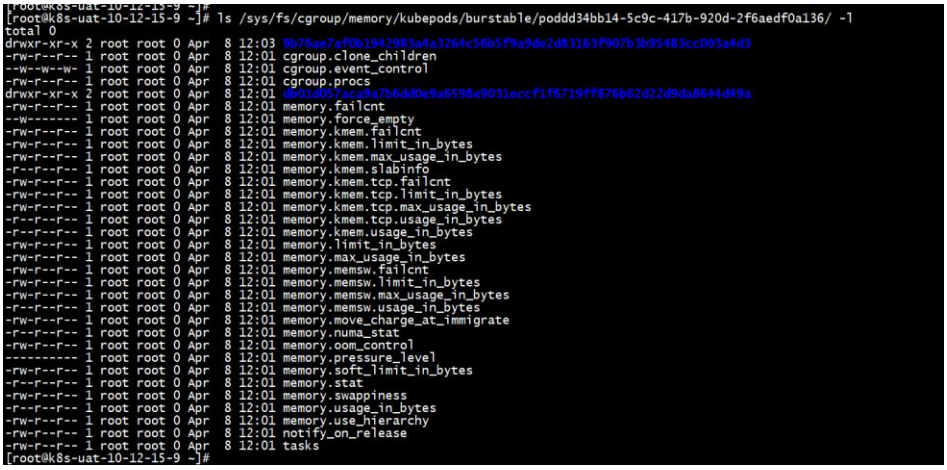
k8s pod 监控分析

k8s 监控问题，经历了几个版本，从 1.12 开始，数据从 k8s metrics api 统一获取，包含核心监控指标和自定义指标，其中核心指标（pod、node 的 cpu、内存、网络、文件等）来自于 kubelet，核心指标还用于 k8s dashboard、hpa 等。这里只说核心指标：

核心指标中和 pod、node 相关的都是由 kubelet metrics api 提供 ，kubelet 内置了 cAdvisor，cAdvisor 调用的是 libcontainer 库，libcontainer 库其实是对 cgroup 文件的封装，所以 cAdvisor 只是个数据转发者，其数据来自于 cgroup 文件，cgroup 文件内容举例如下：

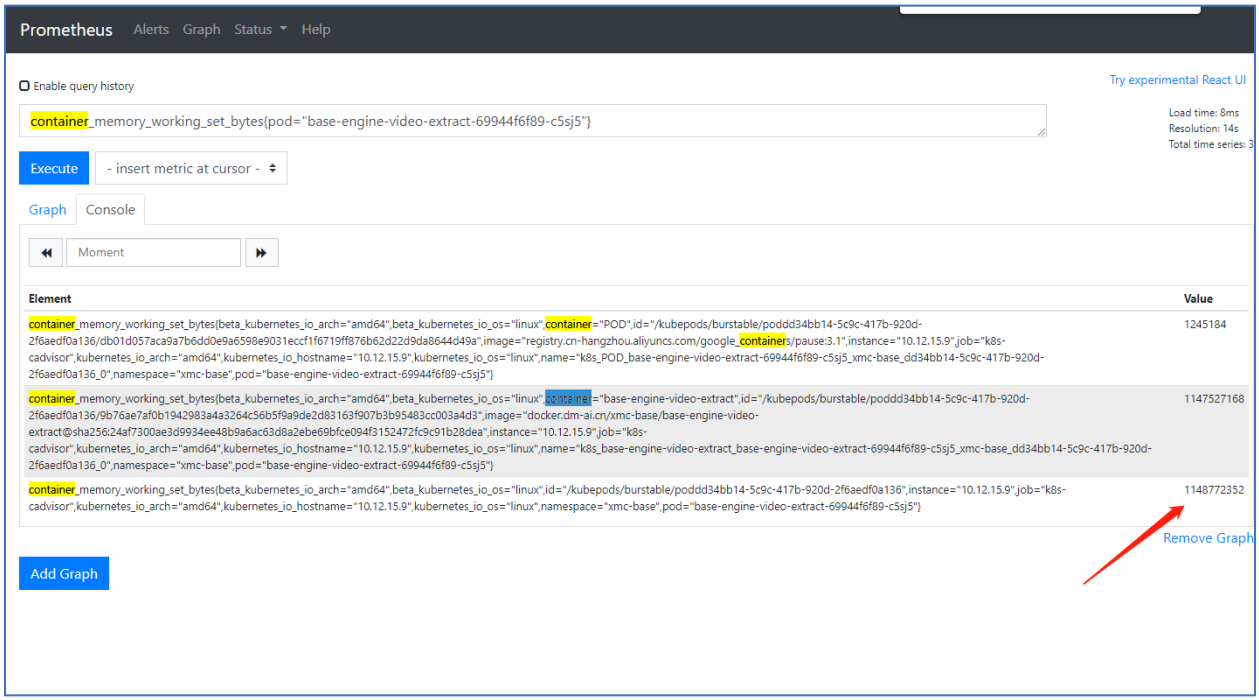


说几个关键的：

cgroup 指标	cgroup 子指标	prometheus 指标	备注
memory.kmem.usage_in_bytes		无	内核空间内存使用
memory.stat	total_inactive_file	无	非活跃内存
	total_active_file	无	活跃内存
	total_rss	container_memory_rss	使用的物理内存，和 docker stats 相同。 docker stats 、top -p $PID 中 RES ， 看到的是这个指标。
	total_cache	container_memory_cache	缓存大小。container_memory_cache =total_inactive_file+total_active_file
memory.usage_in_bytes		container_memory_usage_bytes	内存当前使用量，包含缓存。container_memory_usage_bytes =container_memory_rss + container_memory_cache + memory.kmem.usage_in_bytes

memory.limit_in_bytes		kube_pod_container_resource_limits_memory_bytes	内存限制的使用量
memory.max_usage_in_bytes		container_memory_max_usage_bytes	历史内存内存最大使用量
memory.failcnt		container_memory_failcnt	内存使用量达到限制值的次数
		container_memory_working_set_bytes	内存工作集（working set）使用量，也是 limit 限制时的 oom 判断依据。 kubectl top pod 看到的是这个指标。 container_memory_working_set_bytes = container_memory_usage_bytes - total_inactive_file

下面是一个实例，可以用来验证上述公式，此次的container_memory_working_set_bytes=1148772352




```
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]# cat memory.kmem.usage_in_bytes
10342400
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]#
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]#
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]# cat memory.usage_in_bytes
1148776448
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]#
```

```
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]# cat
memory.stat
cache 0
rss 0
rss_huge 0
mapped_file 0
swap 0
pgpgin 0
pgpgout 0
pgfault 0
pgmajfault 0
inactive_anon 0
active_anon 0
inactive_file 0
active_file 0
unevictable 0
hierarchical_memory_limit 2097152000
hierarchical_memsw_limit 9223372036854771712
total_cache 929800192
total_rss 208633856
total_rss_huge 0
total_mapped_file 0
total_swap 0
total_pgpgin 190186677
total_pgpgout 189908739
total_pgfault 70004868
total_pgmajfault 1
total_inactive_anon 0
total_active_anon 208633856
total_inactive_file 4096
total_active_file 929796096
total_unevictable 0
```

```
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]# cat memory.kmem.usage_in_bytes
10342400
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]#
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]#
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]# cat memory.usage_in_bytes
1148776448
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]#
[root@k8s-uat-10-12-15-9 poddd34bb14-5c9c-417b-920d-2f6aedf0a136]# cat memory.stat
cache 0
rss 0
rss_huge 0
mapped_file 0
swap 0
pgpgin 0
pgpgout 0
pgfault 0
pgmajfault 0
inactive_anon 0
active_anon 0
inactive_file 0
active_file 0
unevictable 0
hierarchical_memory_limit 2097152000
hierarchical_memsw_limit 9223372036854771712
total_cache 929800192
total_rss 208633856
total_rss_huge 0
total_mapped_file 0
total_swap 0
total_pgpgin 190186677
total_pgpgout 189908739
total_pgfault 70004868
total_pgmajfault 1
total_inactive_anon 0
total_active_anon 208633856
total_inactive_file 4096
total_active_file 929796096
total_unevictable 0
```

对于 cgroup 中没有的指标，cAdvisor 是计算出来的，找到相关的源码

container_memory_working_set_bytes = container_memory_usage_bytes - total_inactive_file

对应源码是：

git > go > kubernetes > vendor > github.com > google > cadvisor > container > libcontainer > handler.go

```
554         ret.Memory.Cache = s.MemoryStats.Stats["total_cache"]
555         ret.Memory.RSS = s.MemoryStats.Stats["total_rss"]
556         ret.Memory.Swap = s.MemoryStats.Stats["total_swap"]
557         ret.Memory.MappedFile = s.MemoryStats.Stats["total_mapped_file"]
558     } else {
559         ret.Memory.Cache = s.MemoryStats.Stats["cache"]
560         ret.Memory.RSS = s.MemoryStats.Stats["rss"]
561         ret.Memory.Swap = s.MemoryStats.Stats["swap"]
562         ret.Memory.MappedFile = s.MemoryStats.Stats["mapped_file"]
563     }
564     if v, ok := s.MemoryStats.Stats["pgfault"]; ok {
565         ret.Memory.ContainerData.Pgfault = v
566         ret.Memory.HierarchicalData.Pgfault = v
567     }
568     if v, ok := s.MemoryStats.Stats["pgmajfault"]; ok {
569         ret.Memory.ContainerData.Pgmajfault = v
570         ret.Memory.HierarchicalData.Pgmajfault = v
571     }
572
573     workingSet := ret.Memory.Usage
574     if v, ok := s.MemoryStats.Stats["total_inactive_file"]; ok {      David Ashpole, 3 years ago • update
575         if workingSet < v {
576             workingSet = 0
577         } else {
578             workingSet -= v
579         }
580     }
581     ret.Memory.WorkingSet = workingSet
582 }
583
```