APF 的概念

先看 K8S APF 的官方描述：
“对于集群管理员来说，控制 Kubernetes API 服务器在过载情况下的行为是一项关键任务。kube-apiserver 具有一些可用的控件（即--max-requests-inflight 和--max-mutating-requests-inflight 命令行标志），用于限制将要接受的未完成工作量，从而防止大量入站 API 服务器超载并可能导致 API 服务器崩溃的请求，但这些标志不足以确保最重要的请求在高流量期间通过。

API Priority and Fairness（APF）是一种替代方案，可改善上述 max-inflight 限制。APF 以更细粒度的方式对请求进行分类和隔离。它还引入了数量有限的排队，因此在非常短暂的突发情况下不会拒绝任何请求。 通过使用公平排队技术从队列中分发请求，这样， 一个行为不佳的 K8S apiserver 就不会影响其他 apiserver 接受新的请求（即使优先级相同）。”

简单来说：K8S 为了保护 Api Server 加入了限流机制，避免 Api 被恶意请求消耗资源然后死掉。但目前这个功能存在 bug。

K8S APF bug 的根本原因是：社区里把“SA 访问 workload-low 的加权：20Account Token 访问 global-default 加权:100”两个默认值写的不合理 “SA 访问 workload-low 的加权：20”加权过低 然后 OCP 把 APF 这个 feature 引入了 而 SA 访问 workload-low 的加权” 恰好是 K8S 集群管理流量的的通路。由于加权低，造成队列一大堆 pendding 请求，K8S 集群的体现是出现：KubeAPILatencyHigh,KubeAPIErrorsHigh, KubeAPIDown 等报错。
https://github.com/kubernetes/kubernetes/pull/95259

OCP 中关于 APF 的设置

在 OCP 中，如果发现 KubeAPILatencyHigh, KubeAPIErrorsHigh, KubeAPIDown 的报警问题，使用如下命令进行判断：
oc get --raw /debug/api_priority_and_fairness/dump_priority_levels

```
PriorityLevelName,                 ActiveQueues, IsIdle, IsQuiescing, WaitingRequests, ExecutingRequests
workload-high,                     0,            true,   false,       0,              0
workload-low,                      106,          false,  false,       577,            380
catch-all,                         0,            false,  false,       0,              1
exempt,                            <none>,       <none>, <none>,      <none>,         <none>
global-default,                    0,            true,   false,       0,              0
leader-election,                   0,            true,   false,       0,              0
openshift-control-plane-operators, 0,            true,   false,       0,              0
system,                            0,            true,   false,       0,              0
```

上图中我们看到 workload-low 那一行有很多等待。

针对这个问题，目前在 OpenShift 的方法是暂时关闭这个 APF 的功能：

#oc patch kubeapiserver cluster --type=merge -p
'{"spec":{"unsupportedConfigOverrides":{"apiServerArguments":{"feature-gates":["APIPriorityAndFairness=false"]}}}}'

执行完命令后，关注 openshift-kube-apiserver ns 中的 pod：
oc get pods -n openshift-kube-apiserver -w

kube-api server的pod自动重建（先自动创建 installer 的pod，然后自动重建 api-server pod）。

重建成功：

```
[root@bastion ~]# oc get pods -n openshift-kube-apiserver
NAME                                            READY   STATUS      RESTARTS   AGE
installer-11-master-0.ocp46.ats.com             0/1     Completed   0          4m36s
installer-11-master-1.ocp46.ats.com             0/1     Completed   0          2m9s
kube-apiserver-master-0.ocp46.ats.com           5/5     Running     0          4m24s
kube-apiserver-master-1.ocp46.ats.com           5/5     Running     0          116s
kube-apiserver-master-2.ocp46.ats.com           5/5     Running     0          10d
revision-pruner-10-master-0.ocp46.ats.com       0/1     Completed   0          25m
revision-pruner-10-master-1.ocp46.ats.com       0/1     Completed   0          30m
revision-pruner-10-master-2.ocp46.ats.com       0/1     Completed   0          28m
revision-pruner-11-master-0.ocp46.ats.com       0/1     Completed   0          2m39s
revision-pruner-11-master-1.ocp46.ats.com       0/1     Completed   0          12s
```

需要注意的是，如果要对 OCP 进行升级，请在升级前对这个参数进行还原，以免影响 OCP 的升级。
#oc patch kubeapiserver cluster --type=merge -p
'{"spec":{"unsupportedConfigOverrides":{"apiServerArguments":{"feature-gates":["APIPriorityAndFairness=true"]}}}}'

拓展知识

流控 API 涉及两种资源。PriorityLevelConfigurations 定义隔离类型和可处理的并发预算量，还可以微调排队行为。 FlowSchemas 用于对每个入站请求进行分类，并与一个 PriorityLevelConfigurations 相匹配。

一个 PriorityLevelConfiguration 表示单个隔离类型。每个 PriorityLevelConfigurations 对未完成的请求数有各自的限制，对排队中的请求数也有限制。

oc get prioritylevelconfigurations

```
[root@bastion ~]# oc get prioritylevelconfigurations
NAME              TYPE      ASSUREDCONCURRENCYSHARES   QUEUES   HANDSIZE   QUEUELENGTHLIMIT   AGE
catch-all         Limited   1                          <none>   <none>     <none>             13d
exempt            Exempt    <none>                     <none>   <none>     <none>             13d
global-default    Limited   100                        128      6          50                 13d
leader-election   Limited   10                         16       4          50                 13d
system            Limited   30                         64       6          50                 13d
workload-high     Limited   40                         128      6          50                 13d
workload-low      Limited   20                         128      6          50                 13d
```

当入站请求的数量大于分配的 PriorityLevelConfigurations 中允许的并发级别时， type 字段将确定对额外请求的处理方式。

FlowSchema 匹配一些入站请求，并将它们分配给优先级。 每个入站请求都会对所有 FlowSchema 测试是否匹配， 首先从 matchingPrecedence 数值最低的匹配开始（我们认

为这是逻辑上匹配度最高)， 然后依次进行，直到首个匹配出现。

oc get flowschema

```
[root@bastion ~]# oc get flowschema
NAME                           PRIORITYLEVEL     MATCHINGPRECEDENCE   DISTINGUISHERMETHOD   AGE   MISSINGPL
exempt                         exempt            1                    <none>                13d   False
system-leader-election         leader-election   100                  ByUser                13d   False
workload-leader-election       leader-election   200                  ByUser                13d   False
system-nodes                   system            500                  ByUser                13d   False
kube-controller-manager        workload-high     800                  ByNamespace           13d   False
kube-scheduler                 workload-high     800                  ByNamespace           13d   False
kube-system-service-accounts   workload-high     900                  ByNamespace           13d   False
service-accounts               workload-low      9000                 ByUser                13d   False
global-default                 global-default    9900                 ByUser                13d   False
catch-all                      catch-all         10000                ByUser                13d   False
```

oc explain prioritylevelconfiguration.spec.limited

assuredConcurrencyShares <integer>

`assuredConcurrencyShares` (ACS) configures the execution limit, which is a limit on the number of requests of this priority level that may be exeucting at a given time. ACS must be a positive number. The server's concurrency limit (SCL) is divided among the concurrency-controlled priority levels in proportion to their assured concurrency shares. This produces the assured concurrency value (ACV) --- the number of requests that may be executing at a time --- for each such priority level:

ACV(l) = ceil(SCL * ACS(l) / (sum[priority levels k] ACS(k)))

bigger numbers of ACS mean more reserved concurrent requests (at the expense of every other PL). This field has a default value of 30