博客文章　专业技术　Amir Rawdat of F5　2017-06-12

NGINX 和 NGINX Plus 的速率限制功能

DDoS, *速率限制*, *分布式拒绝服务*

*速率限制*是NGINX 最有用的功能之一，却往往被误解，且未能得到正确配置。此功能可以限制单个用户在给定时长内可发起的 HTTP 请求数量。此类请求可能非常简单，例如网站主页的 `GET` 请求，或登录页面的 `POST` 请求。

速率限制可用于安全目的，例如减缓暴力破解密码的攻击频率。此功能可以将传入的请求速率限制为实际用户的典型值，（通过日志记录）识别目标 URL，从而达到防范 DDoS 攻击的目的。再概括一点说，可以防止上游应用服务器因同时服务过多用户请求而不堪重负。

这篇博文将介绍如何使用 NGINX 进行速率限制，包括基础知识和高级配置。在NGINX Plus 中，速率限制的工作原理相同。

NGINX Plus R16 及更高版本支持"全局速率限制"：无论请求抵达同一集群中的哪个实例，该集群中的 NGINX Plus 实例都会对抵达请求应用统一的速率限制。（集群中的状态共享也可用于NGINX Plus 的其他功能。）有关详细信息，请参阅我们的博客和《NGINX Plus 管理指南》。

NGINX 速率限制的工作原理

NGINX 速率限制功能采用*漏桶算法*。在电信网络和分组交换计算机网络中，该算法被广泛应用于处理有限带宽下的突发流量。此算法可类比为一个水桶，水从顶部倒入，从底部漏出；如果倒水速度超过漏水速度，桶里的水就会溢出。如果套用在请求处理方面，水代表来自客户端的请求，桶代表一个队列，而队列中的请求遵循先进先出 (FIFO) 调度算法等待处理。桶底的漏水代表从缓冲区释放，接受服务器处理的请求，顶部溢出的水代表未处理的弃置请求。

速率限制基本配置

配置速率限制需要使用两个主要指令，`limit_req_zone` 和 `limit_req`，例如：

```
limit_req_zone $binary_remote_addr zone=mylimit:10m rate=10r/s;

server {
    location /login/ {
        limit_req zone=mylimit;

        proxy_pass http://my_upstream;
    }
}
```

`limit_req_zone` 指令用于定义速率限制的参数，而 `limit_req` 在其出现的上下文环境中启用速率限制（在本例中，应用于指向 **/login/**的所有请求）。

`limit_req_zone` 指令通常在 `http` 模块中定义，以便其可用于多个上下文。`limit_req_zone` 指令采用以下三个参数：

- **Key** – 定义应用限制的请求特征。在本例中， Key 是 NGINX 变量 `$binary_remote_addr`，该变量保存着以二进制表示的客户端 IP 地址。这意味着我们将根据第三个参数定义的请求速率对每个独立 IP 地址进行限制。（之所以使用这个变量，是因为它占用的空间少于 客户端 IP 地址的字符串表示 `$remote_addr`）。

- **Zone** – 定义共享内存区域，用于存储每个 IP 地址的状态以及该 IP 地址访问一个请求受限的 URL 的频率。将信息保存在共享内存中意味着可以在 NGINX worker 进程之间共享该信息。Zone的定义分为两部分：zone名称（写为 `zone=` 关键字）和冒号后的zone大小。约 16,000 个 IP 地址的状态信息需占用 1 兆字节空间，因此本例的区域可以存储约 160,000 个地址。

  如果 NGINX 需要添加新条目时存储空间耗尽，则会删除最早的条目。如果释放的空间仍然不足以容纳新条目，NGINX 会返回状态码 `503 (ServiceTemporarily Unavailable)`。此外，为了防止内存耗尽，NGINX 每新建一个条目时，都会删除前 60 秒内未使用的条目，至多删除两条。

- **Rate** – 设置最大请求速率。在本例中，请求速率不能超过每秒 10 次。实际上，NGINX 追踪请求的时间粒度精确到毫秒，因此该限制相当于每 100 毫秒 (ms) 1 个请求。因为我们不允许突发流量（请参阅下一节），这意味着如果一个请求在前一个请求获准后的 100 毫秒内到达，那么它将被拒绝。

`limit_req_zone` 指令设置了速率限制和共享内存区域的参数，但实际上并未起到限制请求速率的作用。要使速率限制生效，还需要使用 `limit_req` 指令，将速率限制应用于特定 `location` 或 `server` 块。在本例中，我们对指向 **/login/** 的请求进行速率限制。

因此，现在对每个独立的 IP 地址做限制，每秒只能发出 10 个 **/login/** 请求——更准确地说，向该 URL 发送请求后的 100 毫秒内不能再次发出请求。

处理突发流量

如果我们在间隔不到 100 毫秒内收到两个请求怎么办？NGINX 将针对第二个请求向客户端返回状态码 `503`。这应该不是我们想要的效果，因为突发流量是应用的正常现象。我们更希望缓冲那些超限的请求，并及时为它们提供服务。这时，我们就要为 `limit_req` 指令添加 `burst` 参数，更新配置如下：

```
location /login/ {
    limit_req zone=mylimit burst=20;

    proxy_pass http://my_upstream;
}
```

`burst` 参数用于定义客户端在该区域定义的速率上限之外还可以发出的请求数量（以 **mylimit** 区域为样本，速率限制为每秒 10 个请求，即每 100 毫秒 1 个请求）。在前一个请求 后100 毫秒内到达的请求会加入队列，这里我们将队列容量设置为 20。

这意味着，如果来自某一 IP 地址的 21 个请求同时到达，NGINX 会立即将第一个请求转发到上游服务器组，并将其余 20 个请求加入队列。此后，每 100 毫秒转发一个队列中的请求。仅当接收的请求导致队列中的请求数超过 20 时，才会向客户端返回 `503`。

无延迟队列

在配置中加入 `burst` 参数能确保流量平稳，但不是很实用，因为这样会使网站访问显得很慢。在本例中，队列中第 20 个数据包需等待 2 秒钟才会转发，而此时响应可能已经没有意义了。为了应对这种情况，要在使用 `burst` 参数时添加 `nodelay` 参数：

```
location /login/ {
    limit_req zone=mylimit burst=20 nodelay;

    proxy_pass http://my_upstream;
}
```

使用 `nodelay` 参数后，NGINX 仍然根据 `burst` 参数分配队列位置，并且执行已配置的速率限制，但此时并非以一定的时间间隔转发队列请求，而是当请求“太早”到达时，只要队列中有可用位置，NGINX 就会立即转发该请求。NGINX 将该位置标记为“taken”（已占用），并且在适当时长后（本例为 100 毫秒后）再释放供其他请求使用。

沿用此前的假设，如果有 20 个位置的队列目前为空，有来自某个 IP 的 21 个请求同时到达。NGINX 会立即将 21 个请求全部转发，并将队列中的 20 个位置标记为已占用，此后每 100 毫秒释放 1 个位置。（但如果有 25 个请求同时到达， NGINX 将立即转发其中 21 个请求，将 20 个位置标记为 taken，并拒绝 4 个请求，返回状态码 `503`。）

现在，假设在第一组请求被转发的 101 毫秒后，又有 20 个请求同时到达。该队列中只有 1 个位置被释放，因此 NGINX 会转发 1 个请求，拒绝其他 19 个请求并对其返回状态码 `503`。如果经过501 毫秒后，接收到 20 个新增请求，那么此时有 5 个位置，NGINX 会立即转发 5 个请求，拒绝其余 15 个请求。

当前效果相当于每秒 10 个请求的速率限制。如果要使用速率限制，又不想限制请求的准入间隔，那么可以考虑使用 `nodelay` 参数。

注意：对于大多数部署场景，我们都建议在 `limit_req` 指令中引入 `burst` 和 `nodelay` 参数。

二段式速率限制

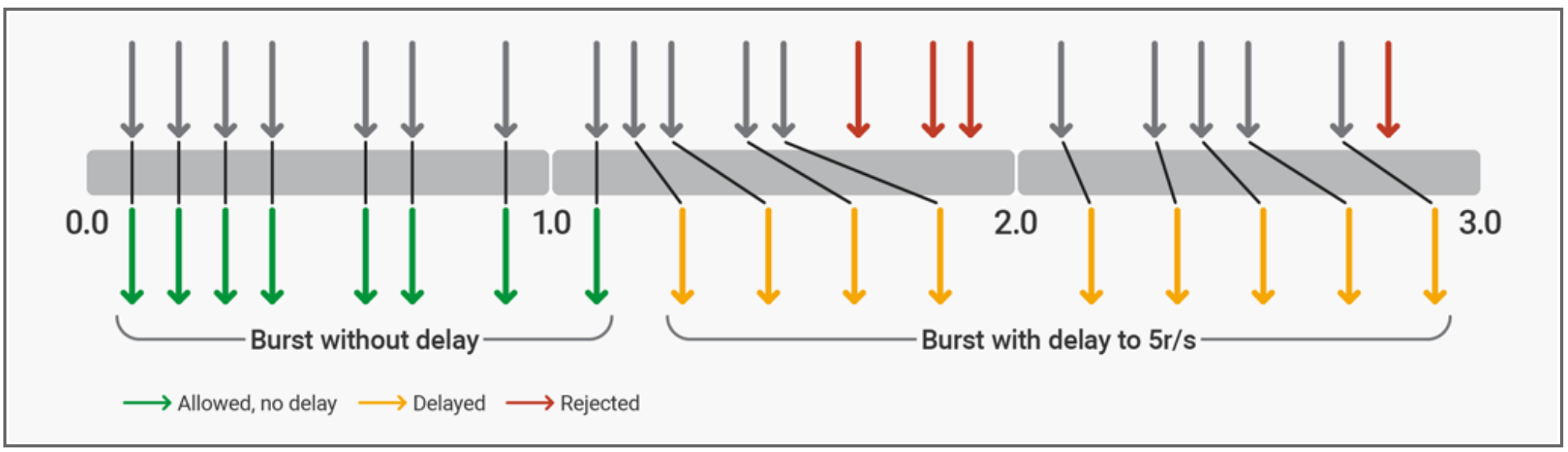
使用 NGINX Plus R17 或 NGINX 开源版 1.15.7，用户可以将 NGINX 配置为允许突发请求，从而适应典型的 Web 浏览器请求模式，然后对额外的请求设置一个临界点，拒绝超过该临界点的额外请求。在 `limit_req` 指令中引入 `delay` 参数可启用二段式速率限制。

为了说明二段式速率限制，我们将 NGINX 配置为限速每秒 5 个请求 (r/s)，以保护网站。该网站每个页面一般有 4-6 个资源，最多不超过 12 个资源。此配置允许的突发请求上限为 12 个，其中前 8 个请求可以立即得到处理。超过 8 个请求后，新增请求将被强制施加 5 r/s 的速率限制。超过 12 个请求后，新增的任何请求都将被拒绝。

```
limit_req_zone $binary_remote_addr zone=ip:10m rate=5r/s;

server {
    listen 80;
    location / {
        limit_req zone=ip burst=12 delay=8;
        proxy_pass http://website;
    }
}
```

`delay` 参数定义了一个点，在突发请求上限内，超出这个点的请求会被限流（延迟），从而达到已定义的速率限制标准。使用此配置，以 8 r/s 速率连续发出请求的客户端会经历如下行为。



速率限制行为图示 `rate=5r/s burst=12 delay=8`

前 8 个请求（`delay` 的赋值）由 NGINX Plus 立即处理，无延迟。接下来的 4 个请求（`burst - delay`差值）被延迟处理，以确保不超过限定的 5 r/s 速率。其余的 3 个请求已超出突发请求上限，因此被拒绝。后续请求会被延迟处理。

高级配置示例

通过组合使用基本速率限制与其他 NGINX 功能，可实现更精细的流量限制。

允许列表

以下示例展示了如何对“允许列表”以外的请求一律施加速率限制。

```
geo $limit {
    default 1;
    10.0.0.0/8 0;
    192.168.0.0/24 0;
}

map $limit $limit_key {
    0 "";
    1 $binary_remote_addr;
}

limit_req_zone $limit_key zone=req_zone:10m rate=5r/s;

server {
    location / {
        limit_req zone=req_zone burst=10 nodelay;

        # ...
    }
}
```

本例同时使用了 geo 和 map 指令。geo 模块为允许列表中 IP 地址的 **$limit** 赋值 **0**，为所有其他 IP 地址的 **$limit** 赋值 **1**。然后，我们使用 map 指令将这些值转换为关键字，例如：

- 如果 **$limit** 值为 **0**，则 **$limit_key** 设为空字符串
- 如果 **$limit** 值为 **1**，则 **$limit_key** 设为二进制格式的客户端 IP 地址

两者组合使用：如果客户端 IP 地址在允许列表中，则 **$limit_key** 设为空字符串；否则设置为该客户端的 IP 地址。当 **limit_req_zone** 目录的第一个参数（key）为空字符串时，不应用限制，因此允许列表中的 IP 地址（在 10.0.0.0/8 和 192.168.0.0/24 子网中）不受限制。而其他 IP 地址一律限速为每秒 5 个请求。

limit_req 指令对 **/** location 限速，但允许无延迟转发最多 10 个超出配置上限的突发请求数据包。

在单一 Location 中使用多个 **limit_req** 指令

用户可以在单一 location 中使用多个 **limit_req** 指令。此时会对符合条件的特定请求应用全部限制，这意味着最严格的一项限制会生效。例如，如果有多项指令施加延迟效果，则最长的延迟生效。同样，如果有任何一项指令的效果是拒绝请求，则该请求将被拒绝，其他指令的准入无效。

继续扩展上文示例，我们可以对允许列表中的 IP 地址应用速率限制：

```
http {
    # ...

    limit_req_zone $limit_key zone=req_zone:10m rate=5r/s;
    limit_req_zone $binary_remote_addr zone=req_zone_wl:10m rate=15r/s;

    server {
        # ...
        location / {
            limit_req zone=req_zone burst=10 nodelay;
            limit_req zone=req_zone_wl burst=20 nodelay;
            # ...
        }
    }
}
```

允许列表中的 IP 地址不符合第一个速率限制条件 (**req_zone**)，但符合第二个条件 (**req_zone_wl**)，因此被限制为每秒 15 个请求。允许列表以外的 IP 地址同时符合两个速率限制条件，因此更严格的一项限制生效：每秒 5 个请求。

配置相关功能

日志记录

默认情况下，NGINX 会记录因速率限制而延迟或弃置的请求，例如：

```
2015/06/13 04:20:00 [error] 120315#0: *32086 limiting requests, excess: 1.000 by zone "mylimit", client: 192.168.1.2,
server: nginx.com, request: "GET / HTTP/1.0", host: "nginx.com"
```

日志条目包含以下字段：

- `2015/06/13 04:20:00` – 日志的录入日期和时间
- `[error]` – 错误等级
- `120315#0` – NGINX worker 的 process ID 和 thread ID，以 `#` 号分隔
- `*32086` – 速率受限的代理连接的 ID
- `limiting requests` – 表明日志条目记录到一次速率限制
- `excess` – 显示此请求超过配置速率的每毫秒请求数
- `zone` – 定义了强制执行速率限制的区域
- `client` – 发出该请求的客户端 IP 地址
- `server` – 服务器的 IP 地址或主机名
- `request` – 客户端发出的实际 HTTP 请求
- `host` – `Host` HTTP 的 header 值

默认情况下，NGINX 的被拒请求记录等级为 `error` 级，如上例中的 `[error]`所示。（NGINX 的被延迟请求记录等级为较低一级，默认为 `warn` 级。）如需更改日志记录等级，可使用 `limit_req_log_level` 指令。下面，我们将被拒请求的记录等级设置为 `warn`：

```
location /login/ {
    limit_req zone=mylimit burst=20 nodelay;
    limit_req_log_level warn;

    proxy_pass http://my_upstream;
}
```

发送至客户端的错误代码

默认情况下，当客户端超过速率限制时，NGINX 会返回状态码 **503**(**Service Temporarily Unavailable**)。可使用 `limit_req_status` 指令设置不同的状态码（本例中为 **444**）：

```
location /login/ {
    limit_req zone=mylimit burst=20 nodelay;
    limit_req_status 444;
}
```

拒绝对特定 Location 的所有请求

如果要拒绝指向特定 URL 的所有请求，而不仅仅是加以限制，请为其配置一个 `location` 块，并使用 `deny` **all** 指令：

```
location /foo.php {
    deny all;
}
```

结语

我们已经介绍了 NGINX 和 NGINX Plus 提供的许多速率限制功能，包括为指向不同位置的 HTTP 请求设置请求速率，以及配置 **burst** 和 **nodelay** 参数等其他速率限制功能。我们还介绍了高级配置，说明了如何对允许列表和拒绝列表的客户端 IP 地址应用不同限制，也解释了如何记录被拒和延迟的请求。