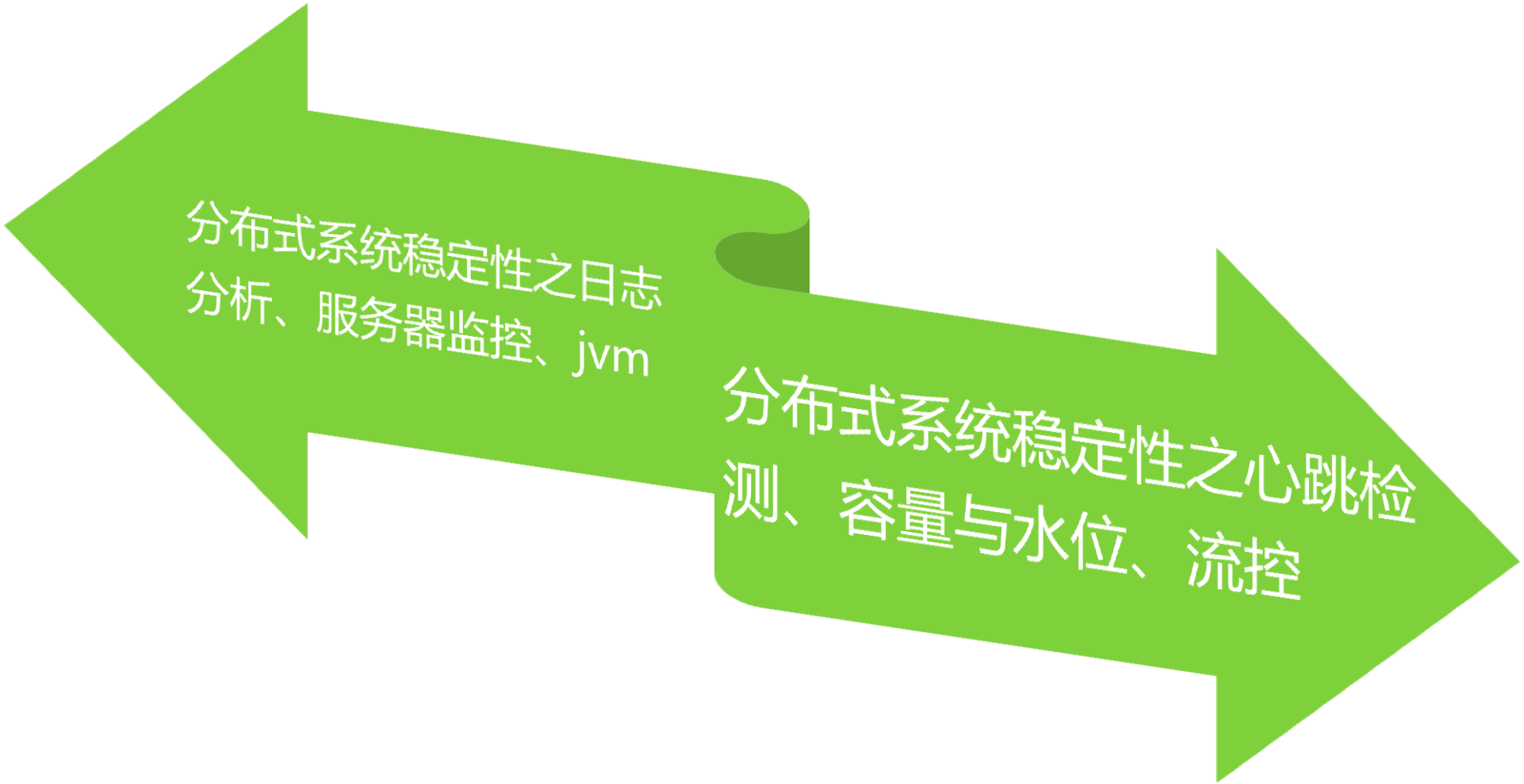
大型电商分布式系统实践 第9周

DATAGURU专业数据分析社区

上节内容回顾



为何要心跳检测

服务器心跳检测--ping

在计算机世界里，ping是最常用的心跳检测方法，通过执行ping命令，发出要求远端主机回应的信息，假如远程主机的网络没有问题，就会对该消息进行回应。ping指令能够检测网络链路是否通畅，远端主机是否能够到达。

```
ping  192.168.2.105
```

服务器心跳检测—应用层检测

通过curl指令定时访问应用中预留的自检url，可以实时的感知应用的健康状态，一旦系统无响应或者是响应超时，即可输出报警信息，以被相应的监控调度系统捕捉到，第一时间通知开发和运维人员进行处理。

通过curl执行自检：

curl http://xxx.com/selfcheck/check.htm

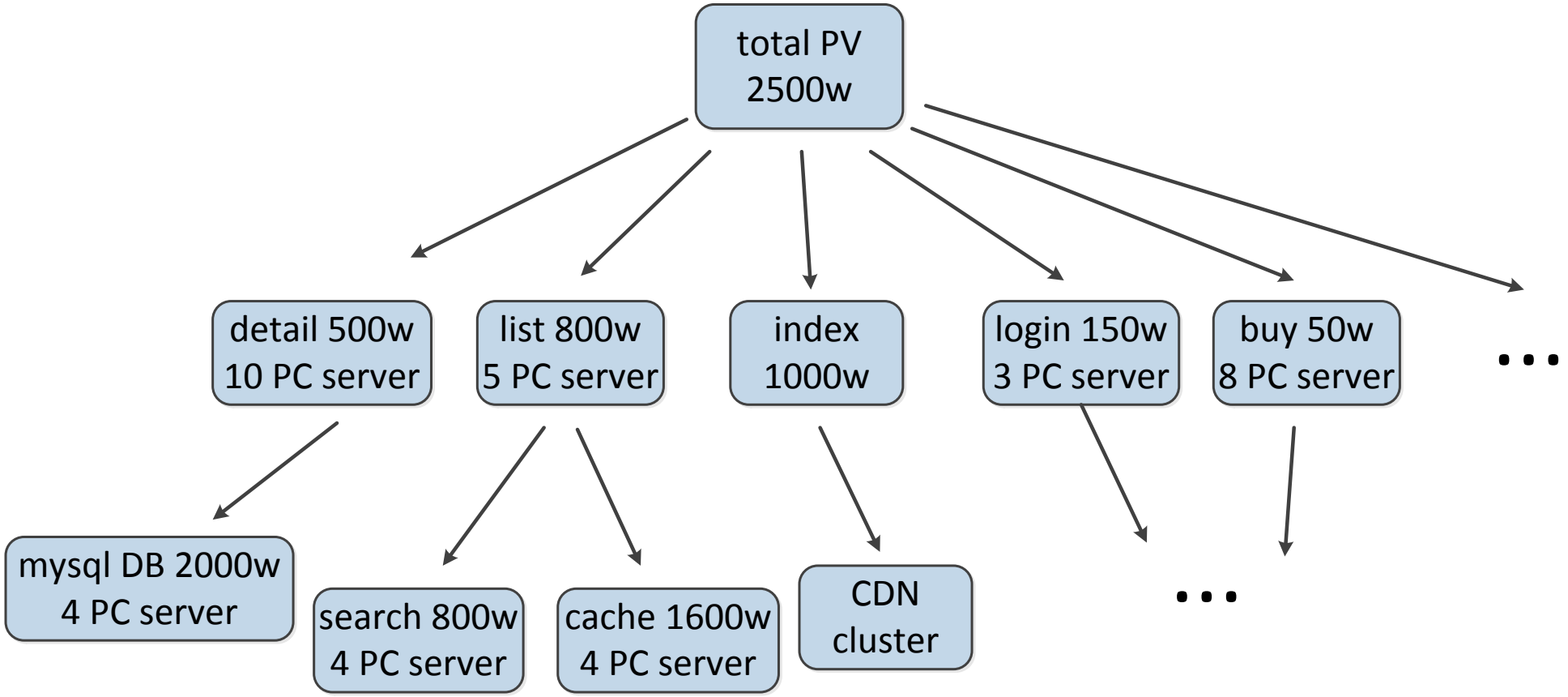
服务器心跳检测—业务检测

假设服务端正常返回，将在response的header中写入如下值：
response.setHeader("Server-Status", "ok");

即使用Server-Status为ok标识服务端当前的返回是正常的，而假如该页面所依赖的服务宕机，页面无法正常显示，则可以不在response的header中设置值，或者设置Server-Status为另外一个值，这样，检测程序便能够感知到系统此时已经发生异常。

通过curl对header进行检测，通过-I选项，只查看header：
curl -I http://xxx.com/selfcheck/index.htm

容量评估—流量分配


total PV
2500w
detail 500w
10 PC server
list 800w
5 PC server
index
1000w
login 150w
3 PC server
buy 50w
8 PC server
...
mysql DB 2000w
4 PC server
search 800w
4 PC server
cache 1600w
4 PC server
CDN
cluster
...

容量评估—机器估算

假设机器的配置都是均衡对称的，我们需要取其中的一个最小子集来进行压力测试，以便得出单个单元所能够承载的访问量。压测机器的配置需要与线上机器保持同等配置，以免结果出现误差。

系统运行的峰值，关乎应用的生死，往往是最后的那一根稻草，将系统压垮。在进行系统峰值评估的时候，一般会遵循一个原则，即所谓的80-20原则，80%的访问请求将在20%的时间内到达，这样，便可以根据系统对应的PV，计算出系统的峰值qps，如果是大型促销或者是推广力度较以往更大，视情况可乘以3-5倍，计算公式如下：

峰值qps = (总的PV * 80%) / (60 * 60 * 24 * 20%)

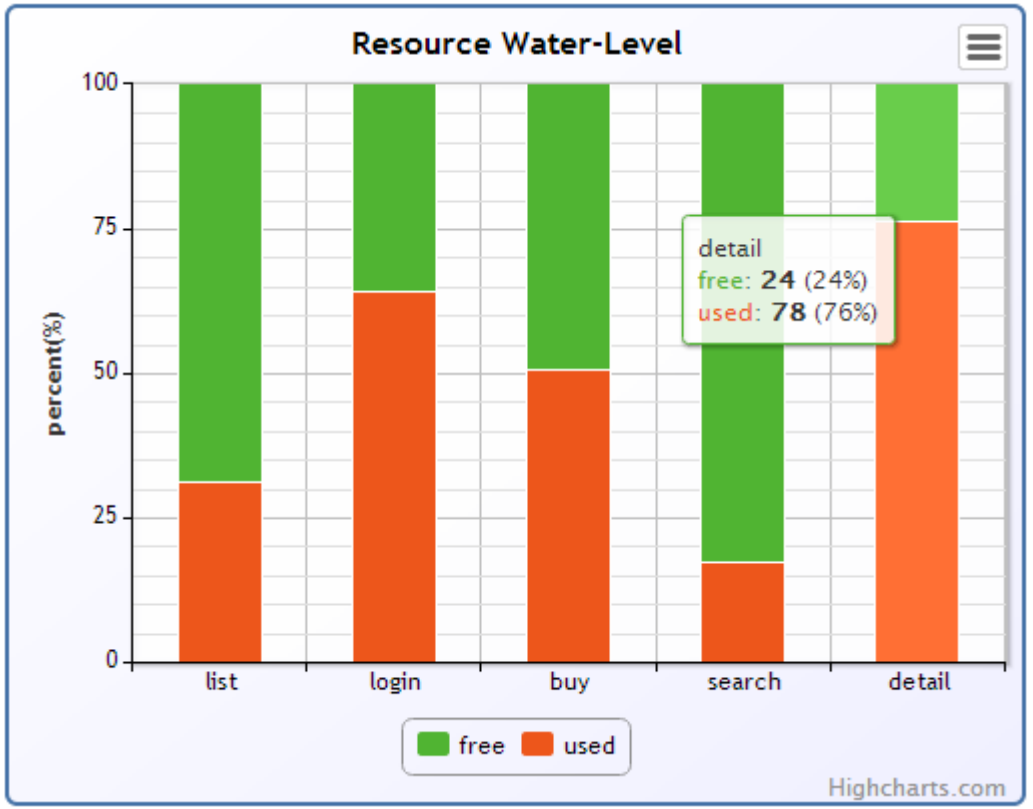
然后，再将总的峰值qps除以单台机器所能够承受的最高的qps值，便是所需要机器的数量：

机器数 = 总的峰值qps / 压测得出的单台机器极限qps

容量评估—应用水位

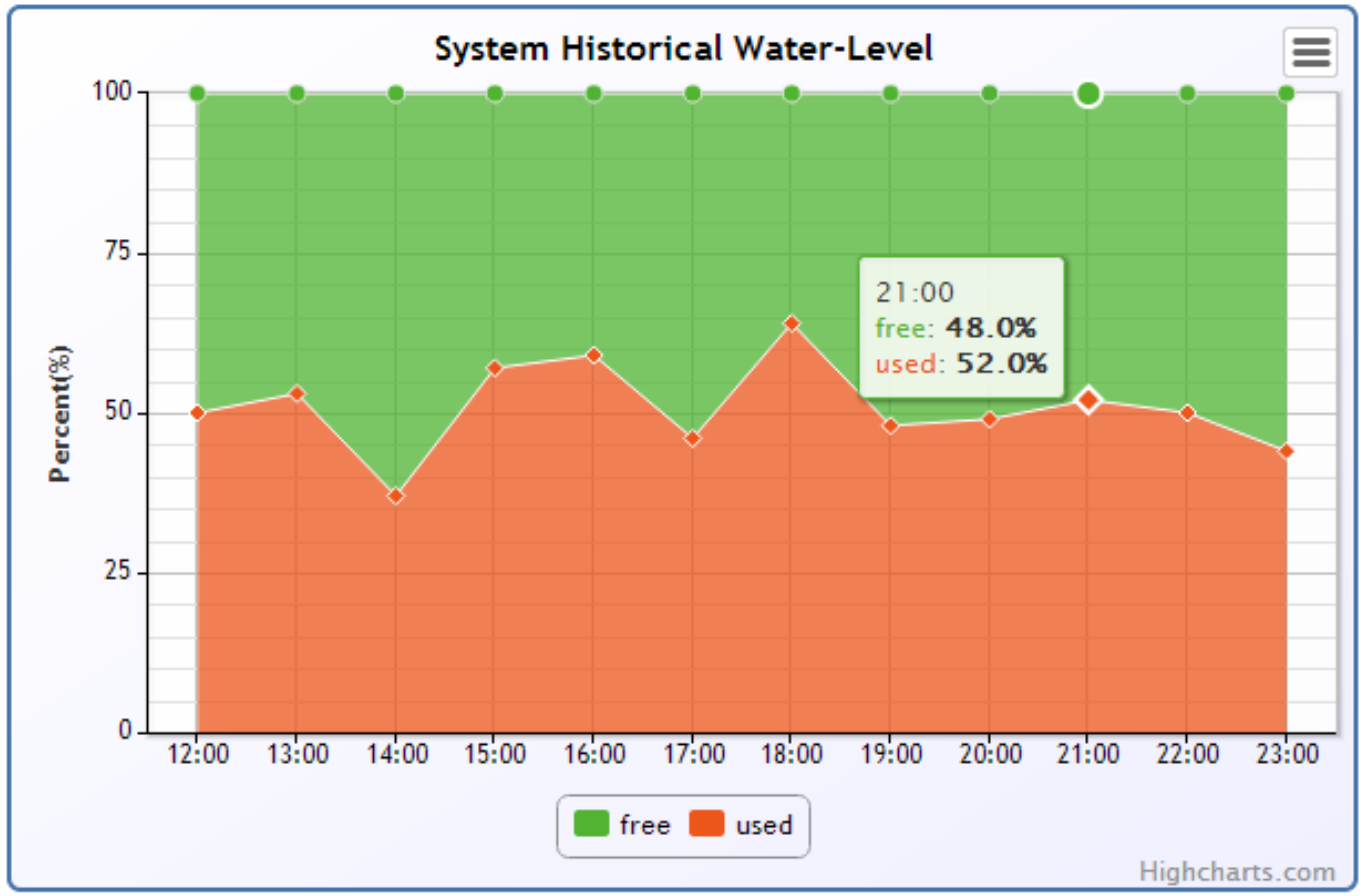
通过访问日志分析，或者其他统计手段，实时计算出当前系统的qps值(前1-2分钟的平均值)，然后，结合系统上线之前压力测试所得到的单台机器的极限qps，乘以当前部署的机器总数，便能够得到当前的水位：

当前水位 = 当前总qps / (单台机器极限qps * 机器数) * 100%



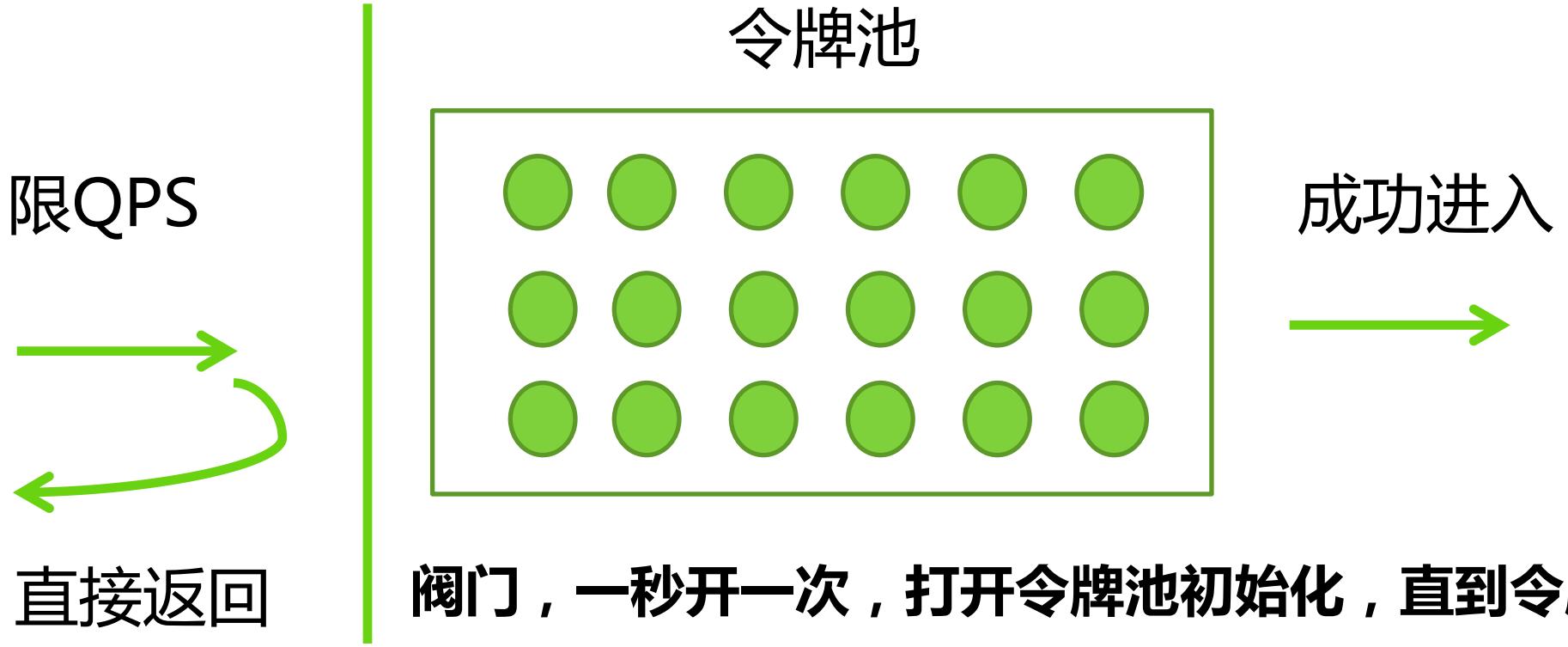
容量评估—历史水位

通过历史数据的积累，还可以绘制出单个系统随着时间推移的历史水位，以便系统管理者能够随时回溯到历史峰值，给容量评估提供参考依据：



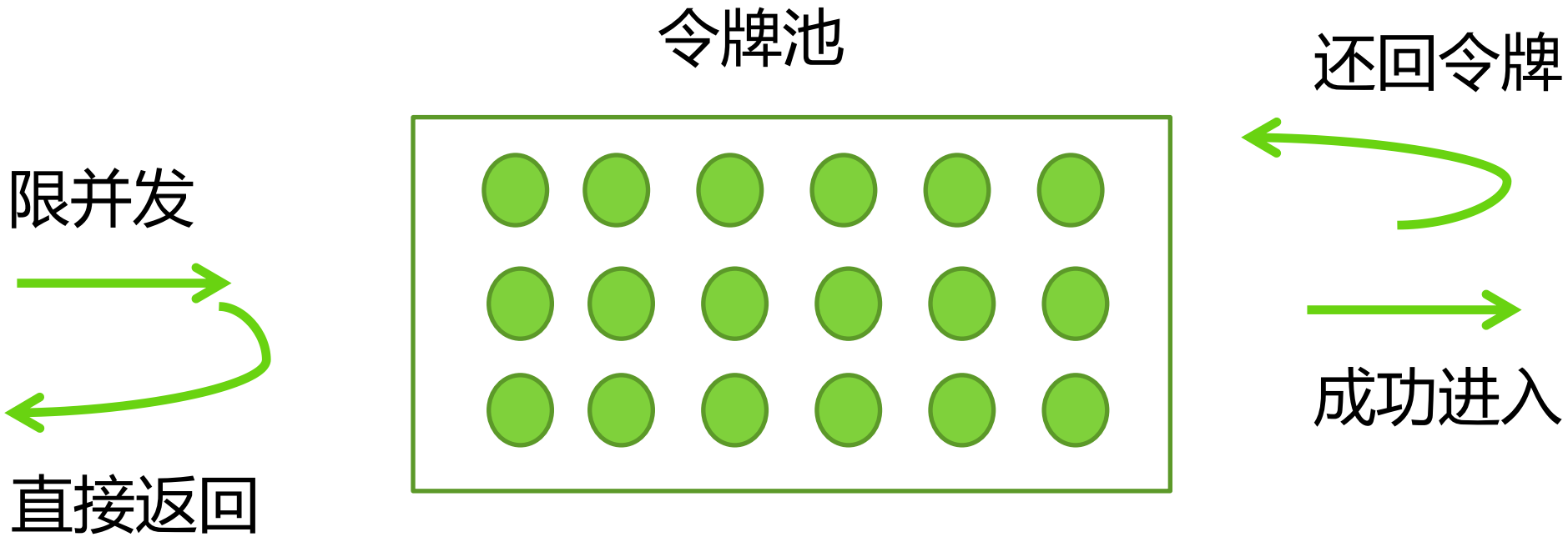
流量控制—为何要流量控制

流量控制—限QPS



阀门，一秒开一次，打开令牌池初始化，直到令牌用完

流量控制—限并发



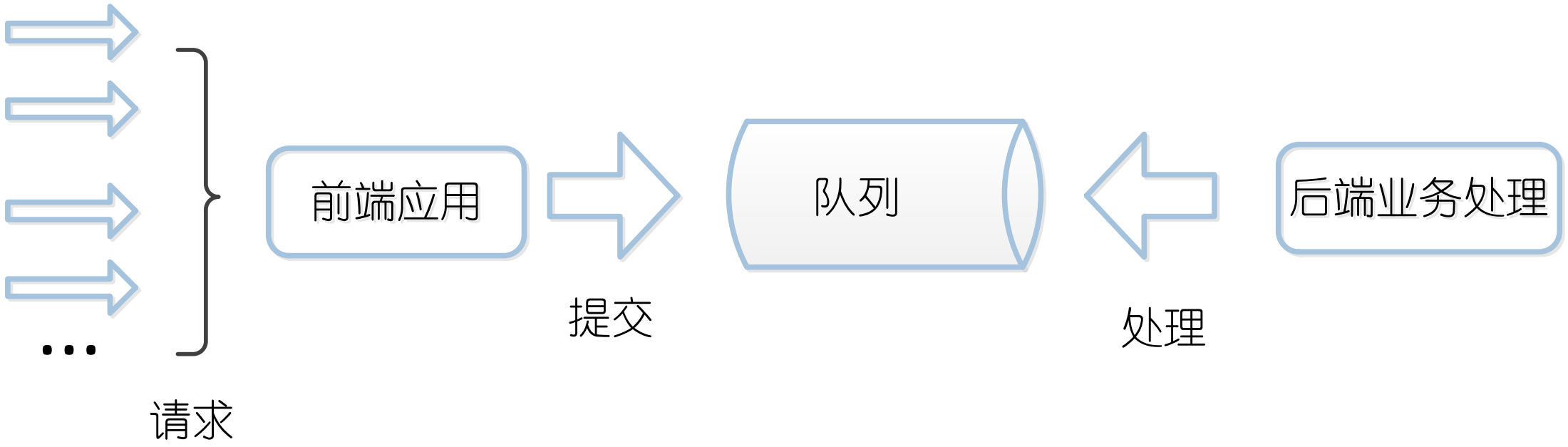
流量控制—java信号量实现流控

```
//信号量定义
Semaphore semphore = new Semaphore(100);

…

//请求处理
if(semphore.getQueueLength() > 0){
        return;
}
try {
        semphore.acquire();

//处理具体的业务逻辑

} catch (InterruptedException e) {
        e.printStackTrace();
}finally{
        semphore.release();//释放
}
```

流量控制—java信号量实现流控

```
//信号量定义
Semaphore semphore = new Semaphore(100);

…

//请求处理
if(semphore.getQueueLength() > 40){
        return;
}
try {
        semphore.acquire();

//处理具体的业务逻辑

} catch (InterruptedException e) {
        e.printStackTrace();
}finally{
        semphore.release();//释放
}
```

流量控制—通过分布式队列来流控



服务稳定性

稳定

Thanks

FAQ时间