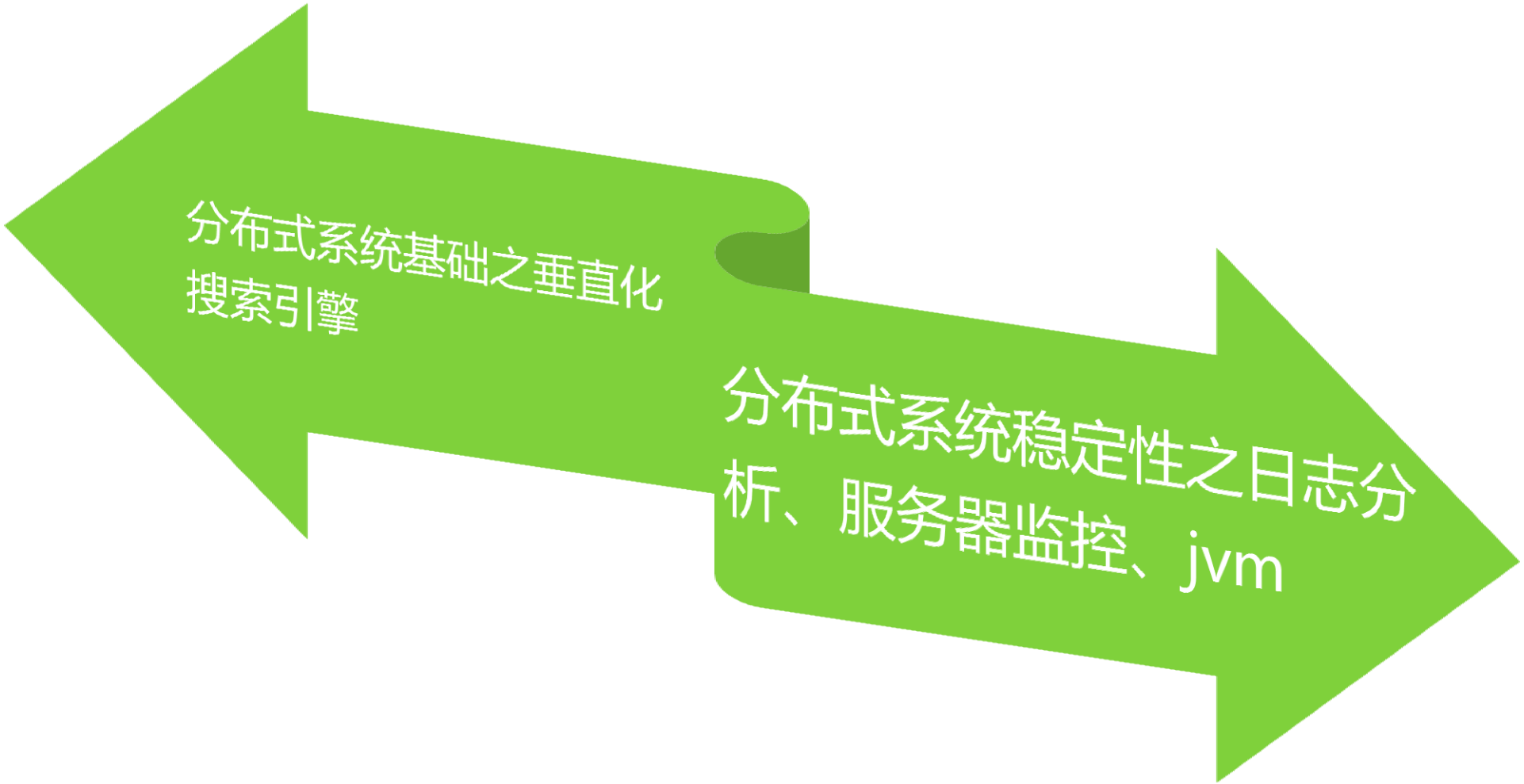
大型电商分布式系统实践 第8周

DATAGURU专业数据分析社区

上节内容回顾



稳定的重要性

一家成熟的大型网站，就如一台时刻不停歇的印钞机，只要它不停止工作，即使不做更新不搞活动，也能够给他的所有者实实在在的带来收益，给它的用户带来价值。一旦哪天印钞机坏了，工作人员应该在第一时间内知晓，并进行修理，因为拖的时间越长，所带来的损失越大。同理，要保障线上系统的安全稳定的运行，开发人员也需要知晓系统当前的运行情况，当发生故障系统不可用时，相关的开发人员也应该第一时间获得消息，进行修复。

日志分析常用命令--cat

cat　　查看文件的内容

cat命令是一个显示文本文件内容的便捷工具，如果一个日志文件比较小，可以直接使用cat命令将其内容打印出来，进行查看，但是，对于较大的日志文件，请不要这样做，打开一个过大的文件可能会占用过多的系统资源，从而影响系统对外的服务。

日志分析常用命令—more or less

more 分页显示文件

cat的缺点在于，一旦执行后，便无法再进行交互和控制，而more命令可以分页的展现文件内容，按enter键显示文件下一行，按空格键便显示下一页，按f键显示下一屏内容，按b键显示上一屏内容。

另一个命令less提供比more更加丰富的功能，支持内容查找，并且能够高亮显示。

日志分析常用命令—tail

tail　显示文件尾

使用tail命令能够查看到文件最后几行，这对于日志文件非常有效，因为日志文件常常是追加写入的，新写入的内容处于文件的末尾位置。

日志分析常用命令—head

head　显示文件头

与tail命令类似，但是不同的是head命令用于显示文件开头的一组行。

日志分析常用命令—sort

sort　内容排序

一个文件中包含有众多的行，经常需要对这些行中的某一列进行排序操作，sort命令的作用便是对数据进行排序。

日志分析常用命令—wc

wc　字符统计

wc命令可以用来统计指定文件中的字符数，字数，行数，并输出统计结果。

日志分析常用命令—uniq

uniq 查看重复出现的行

uniq命令可以用来显示文件中行重复的次数，或者显示仅出现一次的行，以及仅仅显示重复出现的行，并且，uniq的去重针对的只是连续的两行，因此它常常与sort结合起来使用。

日志分析常用命令—curl

curl　URL访问工具

要想在命令行下通过HTTP协议访问网页文档，就不得不用到一个工具，这便是curl，它支持HTTP，HTTPS，FTP，FTPS，Telnet等多种协议，常被用来在命令行下抓取网页和监控WEB服务器状态。

发挥命令的威力—查看请求访问量

对于在线运行的系统来说，常常会碰到各种不怀好意的恶意攻击行为，其中比较常见的便是HTTP flood，也称为CC攻击。如何能够快速的定位到攻击，并迅速响应，便成为开发运维人员必备的技能。定位问题最快捷的办法，便是登录到相应的应用，查看访问日志，找到相应的攻击来源，如访问量排名前10的ip地址：

```
cat access.log | cut -f1 -d " " | sort | uniq -c | sort -k 1 -n -r | head -10
```

页面访问量排名前10的url：

```
cat access.log | cut –f4 -d " " | sort | uniq -c | sort -k 1 -n -r | head -10
```

发挥命令的威力—查看最耗时的页面

对于开发人员来说，页面的响应时间是非常值得关注的，因为这直接关系到用户能否快速的看到他想看到的内容。因此，开发人员常常需要将响应慢的页面找出来，进行优化：

```
cat access.log | sort -k 2 -n -r | head -10
```

发挥命令的威力—统计404页面占比

对于请求的返回码，有些时候也是需要关注的，比如，如果404请求占比过多，要么就是有恶意攻击者在进行扫描，要么就是系统出现问题了，同样，对于500的请求也是如此，可以通过如下命令来查看404请求的占比：

```
export total_line=`wc -l access.log | cut -f1 -d " "` && export
not_found_line=`awk '$6=='404'{print $6}' access.log | wc -l` && expr
$not_found_line \* 100 / $total_line
```

sed编辑器

sed支持在命令行直接指定文本编辑命令，具体格式如下：
sed [options] 'command' file(s)
command为具体的文本编辑命令，而file为输入的文件。

如将日志文件中的xxx替换成yahoo输出：
sed ‘s/xxx/yahoo/’ access.log | head -10

筛选日志中指定的行输出：
sed -n '2,6p' access.log

根据正则表达式删除日志中指定的行：
sed '/qq/d' access.log

显示文件行号：
sed '=' access.log

awk程序

awk使用的通用格式如下：

```
awk  [option]  'pattern {action}'  file
```

其中option为命令的选项，pattern为行匹配规则，action为执行的具体操作，如果没有pattern，则对所有行执行action，而如果没有action，则打印所有匹配的行，file为输入的文件。

打印文件指定的列：

```
awk '{print $1}' access.log | head -10
```

筛选指定的列，并打印出其中一部分行：

```
awk '/google/{print $5,$6}' access.log | head -10
```

对内容进行格式化输出：

```
awk '{line = sprintf ( "method:%s,response:%s", $3 , $7 ); print line}'
access.log | head -10
```

awk程序—计算页面的平均响应时间

```
{
  if( map[$4] > 0 ){
    map[$4]=map[$4] + $2
    map_time[$4]=map_time[$4] + 1
  }
  else{
    map[$4]=$2
    map_time[$4]= 1
  }
}
END{
  for(i in map){
    print i"="map[i]/map_time[i];
  }
}
```

终极武器—shell脚本

脚本能够更加方便的使用外部工具和命令，将内容输出到各个通道甚至是数据库，处理和调度各种复杂的任务等等。
举个例子来说，线上环境常常需要编写一些脚本，来查看系统运行的情况，并且定期执行，一旦系统出现异常，便打印出错误消息，监控系统通过捕捉错误消息，发出报警，下面一个脚本将能够查看系统的load和磁盘占用，在load超过2或者磁盘利用超过85%的情况下报警：

```
load=`top -n 1 | sed -n '1p' | awk '{print $11}'`
load=${load%\,*}
disk_usage=`df -h | sed -n '2p' | awk '{print $(NF - 1)}'`
disk_usage=${disk_usage%\%*}
overhead=`expr $load \> 2.00`
if [ $overhead -eq 1 ];then
 echo "system load is overhead"
fi
if [ $disk_usage -gt 85 ];then
 echo "disk is nearly full, need more disk space"
fi
exit 0
```

集群监控指标--load

系统的load被定义为特定时间间隔内运行队列中的平均线程数。

load的值越大，也就意味着系统的CPU越繁忙，这样线程运行完以后等待操作系统分配下一个时间片段的时间也就越长。一般来说，只要每个CPU当前的活动线程数不大于3，我们认为它的负载是正常的，如果每个CPU的线程数大于5，则表示当前系统的负载已经非常高了，需要采取相应的措施来降低系统的负载，以便影响系统的响应速度。

命令：

top

uptime

集群监控指标—CPU利用率

在linux系统下，CPU的时间消耗主要在这几个方面，即用户进程、内核进程、中断处理、IO等待、nice时间、丢失时间、空闲等几个部分，而CPU的利用率则为这些时间所占总时间的百分比，通过CPU的利用率，能够反映出CPU的使用和消耗情况。
可以通过top命令，查看linux系统的CPU消耗情况：

top | grep Cpu

集群监控指标—磁盘剩余空间

磁盘剩余空间也是一个非常关键的指标，如果磁盘没有足够的剩余空间，正常的日志写入以及系统IO都将无法进行。
通过df命令，能够看到磁盘的剩余空间：

```
df -h
```

集群监控指标—网络traffic

对于对外提供服务的网络应用而言，网络的traffic也很值得关注。一般而言，托管在运营商机房的机器，网络带宽一般情况下不会成为瓶颈，并且集群内部网络都是通过光纤来进行通信，传输质量相当好。但是，对于单个节点来说，由于业务赋予的使命不尽相同，比如某些节点是进行负载均衡和反向代理的节点，某些节点是集群的master，对于网卡和带宽的要求更高，并且，某些极端情况下，比如大促活动，热点事件，网络流量急剧上升，也不排除某些应用会出现网络瓶颈，因此，关注网络的流量，清楚各个节点的阀值和水位，对于开发和运维人员来说，也十分重要。
通过sar命令，可以看到系统的网络状况：

sar –n DEV 1 1

集群监控指标—磁盘IO

磁盘I/O的繁忙度，也是一个重要的系统指标，对于I/O密集型的应用来说，比如数据库应用和分布式文件系统等等，I/O的繁忙程度也一定程度的反映了系统的负载情况，容易成为应用性能的瓶颈。
查看系统的I/O状况：

```
iostat -d -k
```

集群监控指标—内存使用

程序运行时的数据加载，线程并发，I/O缓冲等等，都依赖于内存，可用内存的大小，决定了程序是否能正常运行以及运行的性能。
通过free命令，能够查看到系统的内存使用情况，加上-m参数表示以M为单位：

```
free -m
```

集群监控指标—qps

qps是query per second的缩写，即每秒查询数，qps很大程度上代表了系统在业务上的繁忙程度，而每次请求的背后，可能对应着多次磁盘I/O，多次网络请求，以及多个CPU时间片。

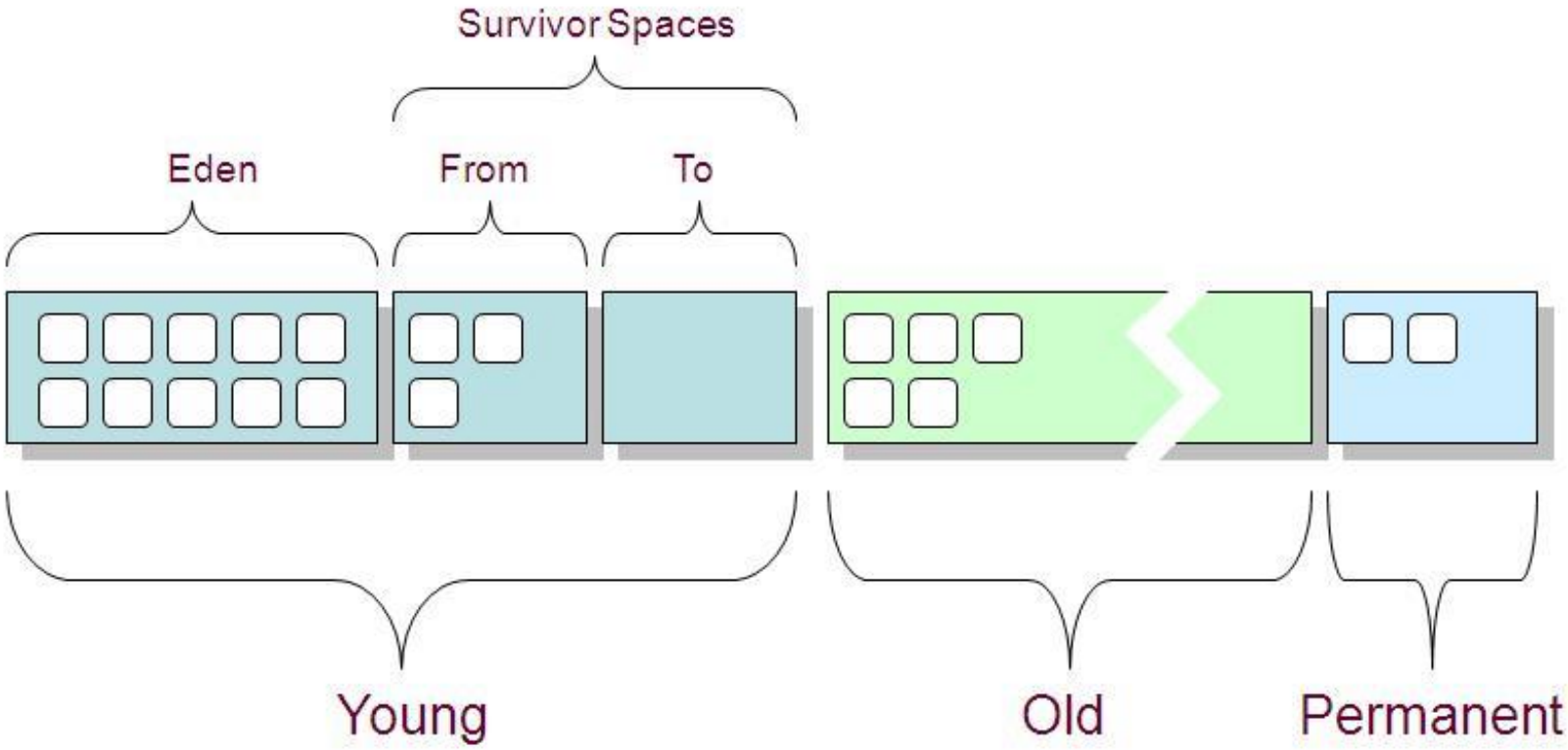
通过关注系统的qps数，我们能够非常直观的了解到当前系统业务情况，一旦当前系统的qps值超过所设置的预警阀值，即可考虑增加机器以对集群进行扩容，以免因压力过大而导致宕机，集群预警阀值的设置，可以根据前期压测得出的值，综合后期的运维经验，评估一个较为合理的数值。

集群监控指标—rt

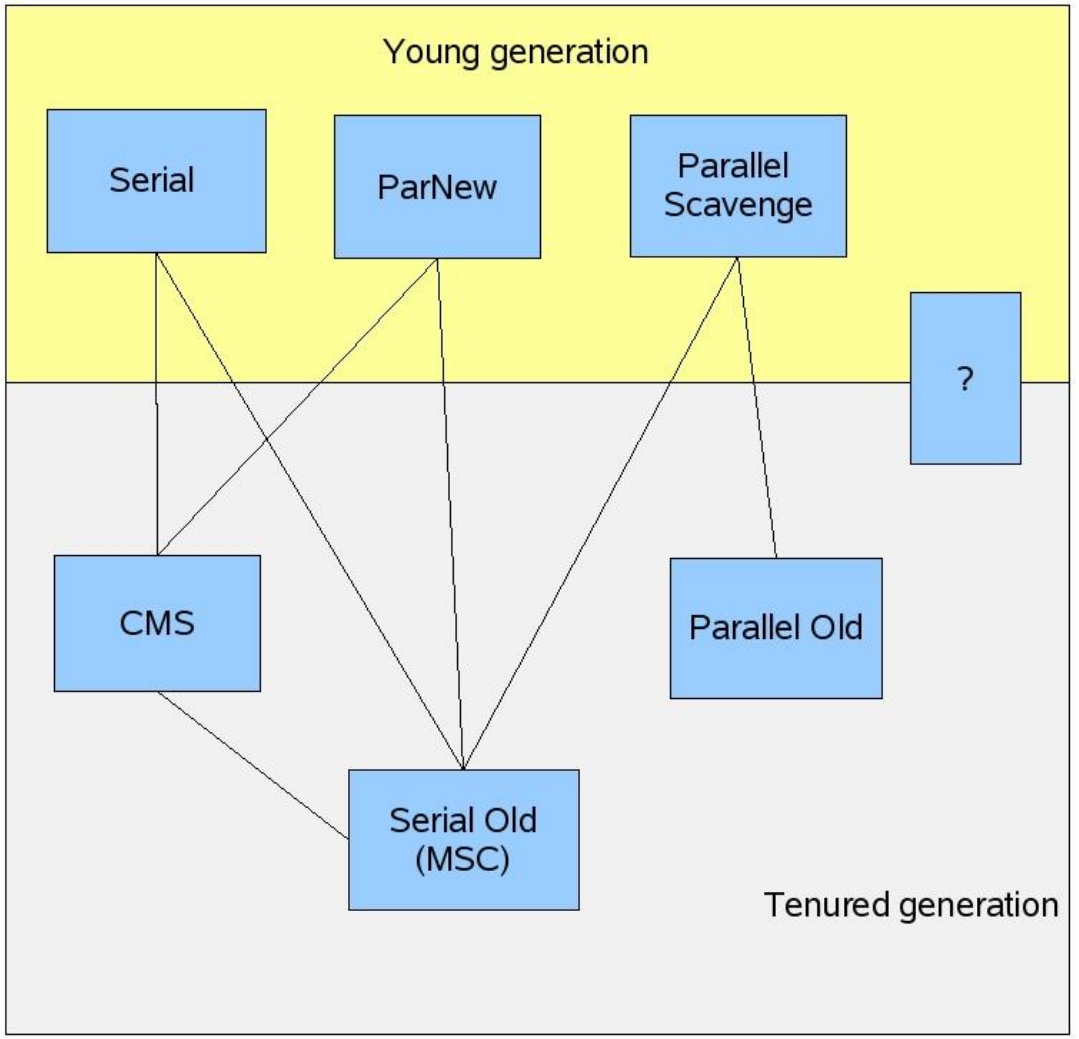
rt是response time的缩写，即请求的响应时间。响应时间是一个非常关键的指标，直接关系到前端的用户体验，因此，任何开发人员和设计师都想尽可能的降低系统的rt时间。对于WEB应用来说，如果响应太慢而导致用户失去耐心，将流失大量的用户。降低rt时间需要从各个方面入手，找到应用的瓶颈，对症下药，比如，通过部署CDN边缘节点，缩短用户请求的物理路径，通过内容压缩，减少传输的字节数，使用缓存，减少磁盘I/O和网络请求等等。

通过apache或者nginx的访问日志，便能够得知每个请求的响应时间。以nginx为例，访问日志的输出格式中，增加$request_time的输出，便能够获得响应时间。

集群监控指标—java的GC



集群监控指标—java的GC



集群监控指标—java的GC日志

Parallel Scavenge收集器的GC日志的格式如下：

2014-02-17T14:17:19.047+0800: 97.756: [GC [PSYoungGen: 1348425K->2096K(2216192K)] 3360755K->2018009K(5116160K), 0.0640020 secs] [Times: user=0.24 sys=0.01, real=0.06 secs]

Parallel Scavenge/Parallel Old组合Full GC所产生日志如下：

2014-02-18T04:37:20.618+0800: 99.327: [Full GC [PSYoungGen: 2779200K->0K(2782336K)] [PSOldGen: 2475753K->1210837K(2899968K)] 5254953K->1210837K(5682304K) [PSPermGen: 94128K->94128K(109568K)], 5.8954740 secs] [Times: user=6.13 sys=0.00, real=5.89 secs]

ParNew收集器的GC日志格式如下所示：

2014-02-20T13:12:23.334+0800: 143.977: [GC 143.977: [ParNew: 2356922K->166185K(2403008K), 0.0275450 secs] 3160143K->1021743K(3975872K), 0.0278960 secs] [Times: user=0.37 sys=0.00, real=0.03 secs]

集群监控指标—java的GC日志

CMS收集器是一款以获取最短回收停顿时间为目的的收集器，它是基于标记-清除算法实现的，运转过程比前面介绍的几个收集器更为复杂，整个过程大致分为四个步骤：

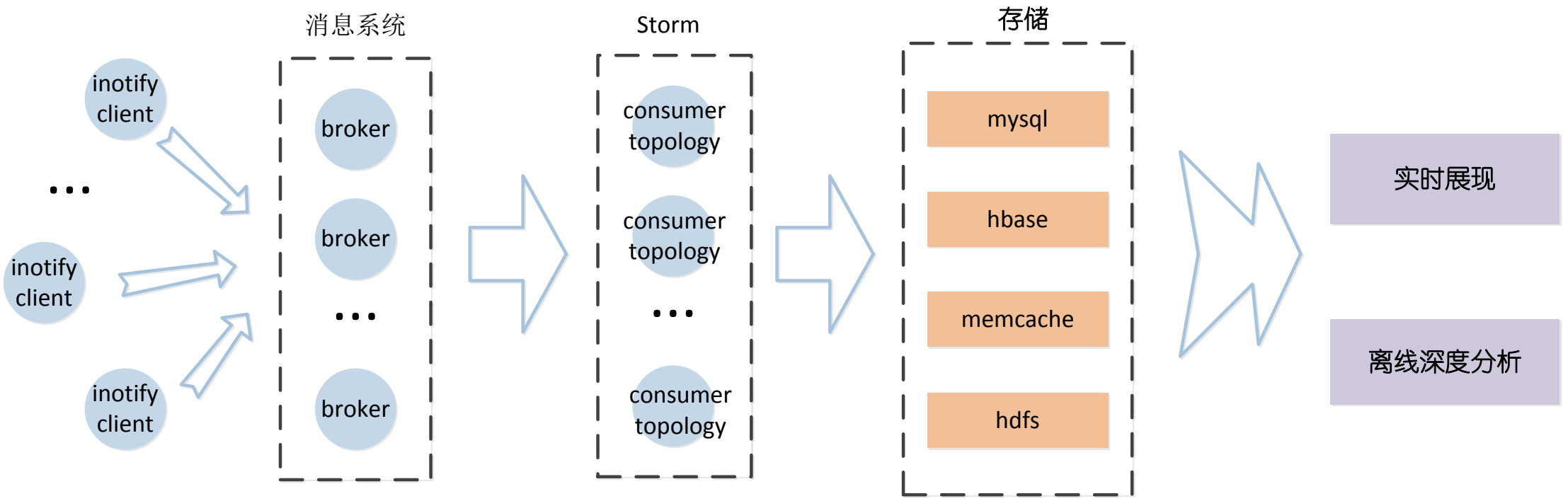
初始标记（CMS initial mark）
并发标记（CMS concurrent mark）
重新标记（CMS remark）
并发清除（CMS concurrent sweep）

2014-02-20T13:11:17.554+0800: 78.196: [GC [1 CMS-initial-mark: 0K(1572864K)] 2240616K(3975872K), 1.1883700 secs] [Times: user=1.19 sys=0.00, real=1.19 secs]
2014-02-20T13:11:18.743+0800: 79.385: [CMS-concurrent-mark-start]
2014-02-20T13:11:18.810+0800: 79.452: [CMS-concurrent-mark: 0.037/0.067 secs] [Times: user=0.37 sys=0.01, real=0.07 secs]
2014-02-20T13:11:18.810+0800: 79.452: [CMS-concurrent-preclean-start]
2014-02-20T13:11:18.838+0800: 79.481: [CMS-concurrent-preclean: 0.027/0.028 secs] [Times: user=0.11 sys=0.00, real=0.02 secs]

集群监控指标—java的GC日志

2014-02-20T13:11:18.838+0800: 79.481: [CMS-concurrent-abortable-preclean-start]
2014-02-20T13:11:19.180+0800: 79.822: [GC 79.822: [ParNew: 2346280K->218432K(2403008K), 0.2444410 secs] 2346280K->263080K(3975872K), 0.2446420 secs] [Times: user=1.53 sys=0.23, real=0.24 secs]
2014-02-20T13:11:23.395+0800: 84.037: [CMS-concurrent-abortable-preclean: 4.172/4.557 secs] [Times: user=15.22 sys=0.39, real=4.56 secs]
2014-02-20T13:11:23.396+0800: 84.038: [GC[YG occupancy: 1337142 K (2403008 K)]84.038: [Rescan (parallel) , 0.0550880 secs]84.094: [weak refs processing, 0.0000140 secs]84.094: [class unloading, 0.0126960 secs]84.106: [scrub symbol & string tables, 0.0146290 secs] [1 CMS-remark: 44648K(1572864K)] 1381790K(3975872K), 0.0851600 secs] [Times: user=0.99 sys=0.00, real=0.09 secs]
2014-02-20T13:11:23.481+0800: 84.124: [CMS-concurrent-sweep-start]
2014-02-20T13:11:23.513+0800: 84.156: [CMS-concurrent-sweep: 0.031/0.032 secs] [Times: user=0.06 sys=0.01, real=0.03 secs]
2014-02-20T13:11:23.514+0800: 84.156: [CMS-concurrent-reset-start]
2014-02-20T13:11:23.530+0800: 84.172: [CMS-concurrent-reset: 0.016/0.016 secs] [Times: user=0.02 sys=0.01, real=0.01 secs]

集群监控指标—数据收集


消息系统
Storm
存储
inotify client
inotify client
inotify client
broker
broker
broker
consumer topology
consumer topology
consumer topology
mysql
hbase
memcache
hdfs
实时展现
离线深度分析

Thanks

FAQ时间