**点评Cat报表相关整理**

# 报表介绍

CAT提供以下几种报表：

（1）、Transaction报表

一段代码运行时间、次数，比如URL、Cache、SQL执行次数和响应时间

（2）、Event报表

一行代码运行次数，比如出现一个异常

（3）、Problem报表

根据Transaction/Event数据分析出来系统可能出现的异常，包括访问较慢的程序等

（4）、Heartbeat报表

JVM内部一些状态信息，比如Memory，Thread等

（5）、Business报表

业务监控报表，比如订单指标。与Transaction、Event、Problem不同，Business更偏向于宏观上的指标，另外三者偏向于微观代码的执行情况

# Transaction报表

## 1、主要功能

监控一段代码运行情况：运行次数、QPS、错误次数、失败率、响应时间统计（平均响应时间、Tp分位值）等等

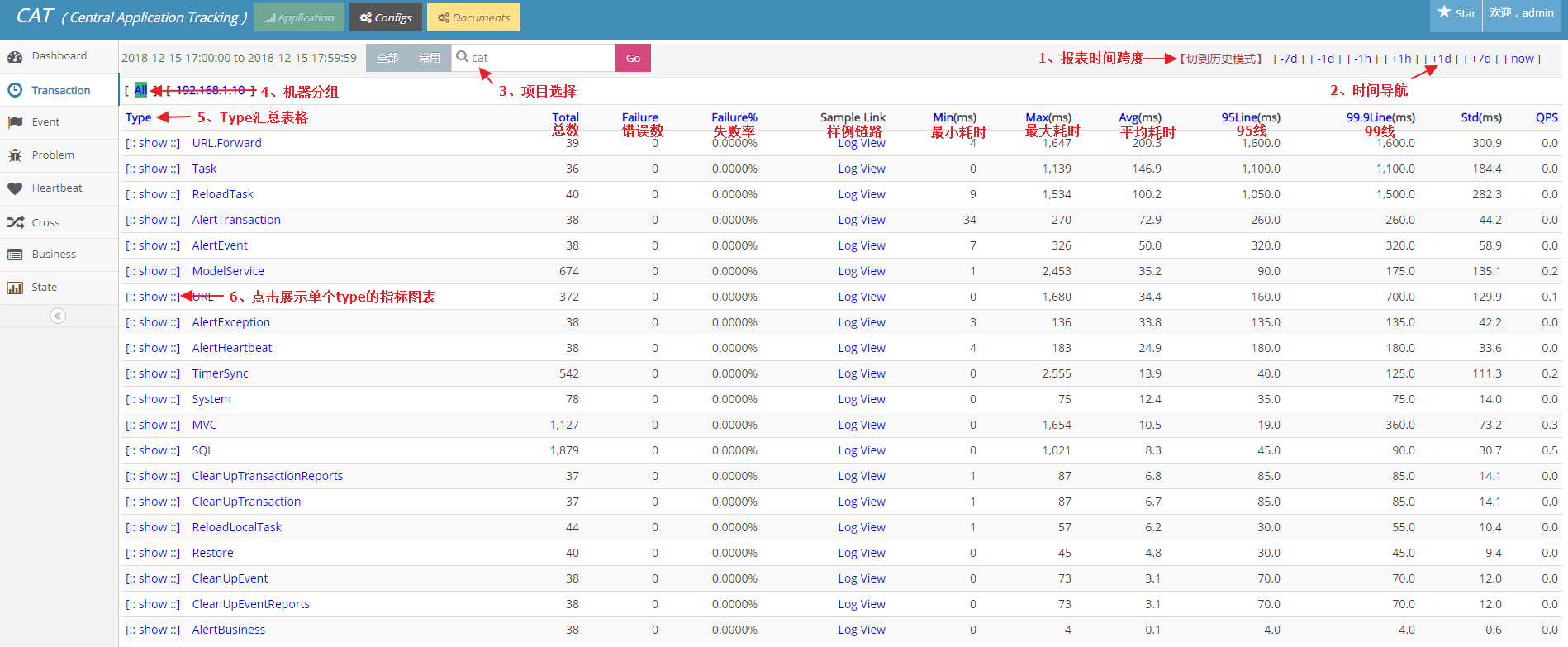
应用启动后默认会打点的部分：

|  |  |  |
| --- | --- | --- |
| 打点 | 来源组件 | 描述 |
| System | cat-client | 上报监控数据的打点信息 |
| URL | 需要接入cat-filter | URL访问的打点信息 |

## 2、报表介绍

### （1）、第一级分类（Type统计界面）

Type统计界面展示了一个Transaction的第一层分类视图，可以知道这段时间里面一个分类运行的次数，平均响应时间，延迟，以及分位线。



从上而下分析报表：

#### A、报表的时间跨度

CAT默认是以一小时为统计时间跨度，点击[切换历史模式]，更改查看报表的时间跨度：默认是小时模式；切换为历史模式后，右侧快速导航变成如下图所示



可以点击进行查看，注意报表的时间跨度会有所不同

#### B、时间选择

通过右上角时间导航栏选择时间，点击[+1h]/[-1h]切换时间为下一小时/上一小时;点击[+1d]/[-1d]切换时间为后一天的同一小时/前一天的同一小时;点击右上角[+7day]/[-7day]切换时间为后一周的同一小时/前一周的同一小时;点击[now]回到当前小时。

#### C、项目选择

输入项目名，查看项目数据;如果需要切换其他项目数据,输入项目名,回车即可。

#### D、机器选择

CAT可以将诺干个机器,作为一个分组进行数据统计。默认会有一个[All]分组,代表所有机器的统计数据,即集群统计数据。

#### E、所有Type汇总表格

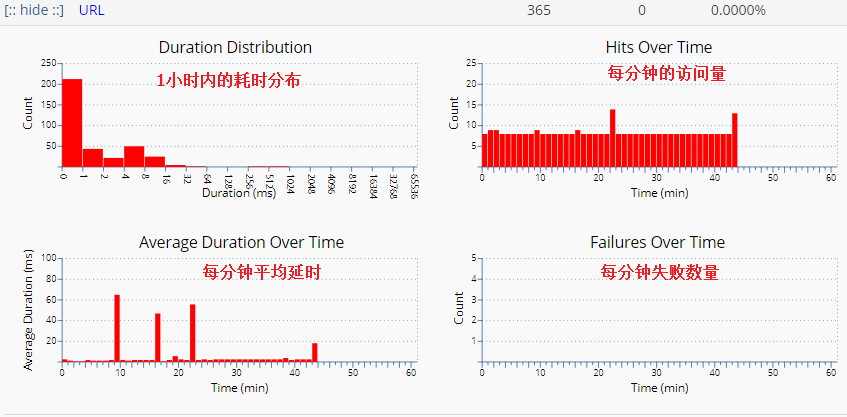
第一层分类（Type）,点击查看第二级分类（称为name）数据：

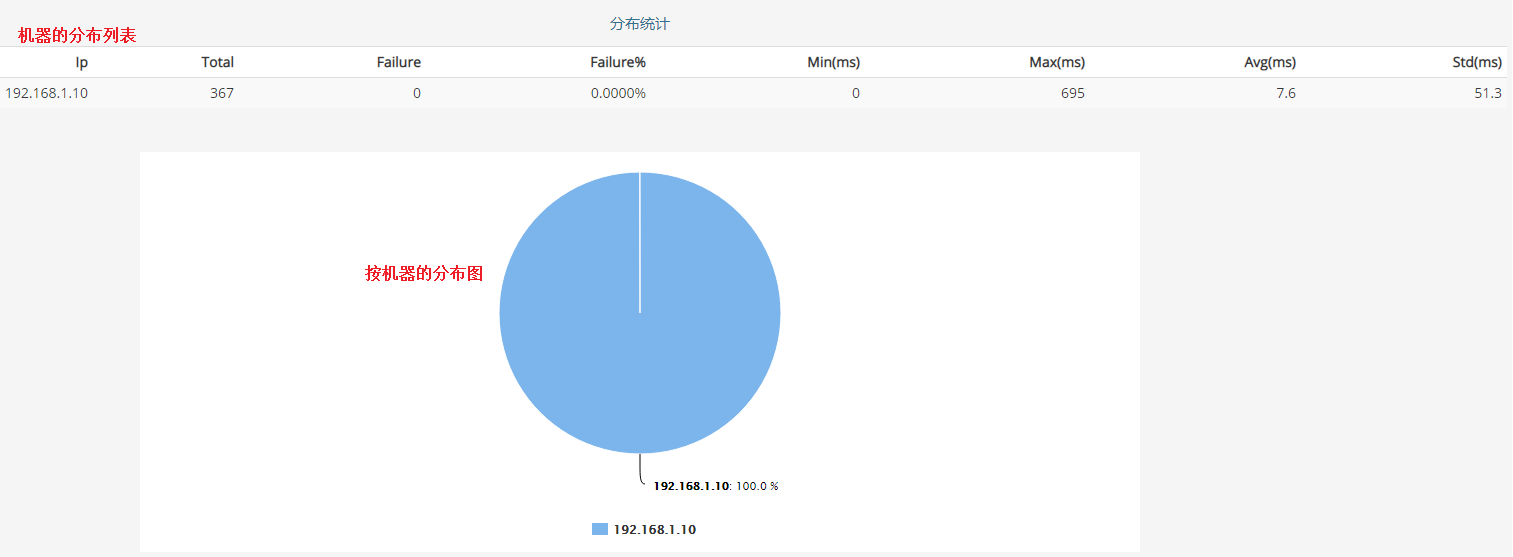
Transaction的埋点的Type和Name由业务自己定义,当打点了Cat.newTransaction(type,name)时,第一层分类是type,第二级分类是name。

第二级分类数据就是统计相同type下的所有name数据,数据均与第一级（type）一样的展示风格。

#### F、单个Type指标图表

点击show,查看Type所有name分钟级统计，如下图:





#### G、指标说明

显示的是小时粒度第一级分类（type）的次数、错误数、失败率等数据

#### H、样本logview

L代表logview，为一个样例的调用链路

#### I、分位线说明

小时粒度的时间第一级分类（type）相关统计

95line：表示95%的请求的响应时间比参考值要小,999line表示99.9%的响应时间比参考值要小,95line以及99line，也称之为tp95、tp99。

## 3、Transaction历史报表

Transaction历史报表支持每天、每周、每月的数据统计以及趋势图，点击导航栏的切换历史模式进行查询。Transaction历史报表以响应时间、访问量、错误量三个维度进行展示，以天报表为例：选取一个type，点击show，即可查看天报表。

## 4、最佳实践

TP分位线计算逻辑

（1）、CAT基于全量数据，在单台机器的上单个小时内部，做全量的区间分桶，在区间分桶也做了部分优化，时间越大桶的区间粒度越大，最后根据count找出分位。

（2）、CAT内部是自己实现的模型和存储，本身支持增加合并和计算，分位线这类并不是适合增量计算，必须依赖全量数据才可以，但是全量的数据存储量非常之大。Transaction报表目前准确统计小时范围单台机器级别分钟级的TP数据。如果看小时级别ALL的统计，是根据当前小时所有的机器进行合并的结果，主要一段时间内所有的机器响应时间波动基本一致，小时级别基本也准确，但是如果个别机器抖动特别大，这个所有的统计会失真。

（3）、CAT后端天、周、月级别的数据是就小时数据的合并，主要用了加权来做，如果一天整个项目的99的波动不大，天级别的99约等于小时级别99。如果一天内的某几个小时的tp99 比较大的话，此做法就会有误差，比如某个服务一天有好几个小时tp99的数据都是正常其他时间段的10倍，这样天级别的99误差就比较大。简而言之天、周、月TP值的统计是一个根据加权算法的近似估计值。

（4）、分位线的统计数据，单台机器小时报表的小时粒度和分钟粒度是完全准确的，天、周、月级别、包括集群的总的分位线都是加权来计算的。计算逻辑如下：

机器A的tp999是timeA，访问总量是countA

机器B的tp999是timeB，访问总量是countB

集群（假设只有A和B两个机器）的tp999的计算公式是：tp999 = (timeA \* countA + timeB \* countB) / (countA + countB)

天、周、月报表计算逻辑类似，只不过把机器之间的运算换成了时间之间的运算

## 5、示例说明

我想监控一段核心代码的执行的情况，能看到统计报表，怎么办？

Step1: 首先，确定监控对象 - 监控某个方法中核心逻辑的健康状况

Step2: 其次，确定监控指标，Transaction统计报表可以提供如下信息 - 每分钟调用次数 - 每分钟的平均响应时间 - 最慢响应时间 - 请求的时间分布

Step3: 然后，进行埋点（前提是已经集成cat-client)

假设监控对象的代码为：

public void shopService() {

service1();

service2();

service3();

}

假设shopService中service3是核心方法，对其进行CAT埋点：

public void shopService() {

service1();

service2();

Transaction transaction = Cat.newTransaction("ShopService", "Service3");

try {

service3();

transaction.setStatus(Transaction.SUCCESS);

} catch (Exception e) {

transaction.setStatus(e); // catch 到异常，设置状态，代表此请求失败

Cat.logError(e); // 将异常上报到cat上

// 也可以选择向上抛出： throw e;

} finally {

transaction.complete();

}

}

# Event报表

## 1、主要功能

监控一段代码运行次数:例如记录程序中一个事件记录了多少次,错误了多少次。Event报表的整体结构与Transaction报表几乎一样,只缺少了响应时间的统计。

应用启动后默认会打点的部分:

|  |  |  |
| --- | --- | --- |
| 打点 | 来源组件 | 描述 |
| System | cat-client | 上报监控数据的打点信息、Reboot重启等 |

## 2、报表介绍

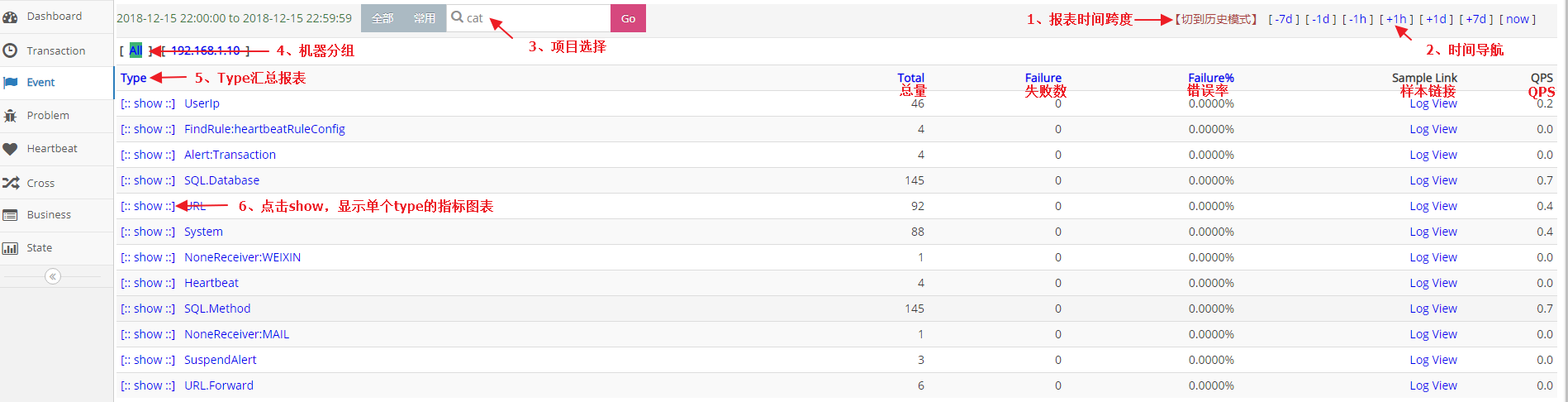
## （1）、第一级分类（Type）统计界面

Type统计界面展示了一个Event的第一层分类的视图，Event相对于Transaction少了运行时间统计。可以知道这段时间里面一个分类运行的次数，失败次数，失败率，采样logView，QPS。

### （2）、第二级分类（Name）统计界面

第二级分类在Type统计界面中点击具体的Type进入，展示的是相同type下所有的name数据，可以理解为某type下更细化的分类。

### （3）、Type统计界面详细介绍



从上而下分析报表

#### A:报表的时间跨度

cat默认是以一小时为统计时间跨度，点击[切到历史模式]，更改查看报表的时间跨度：默认是小时模式；切换为历史模式后，右侧快速导航，变为month(月报表)、week(周报表)、day(天报表)，可以点击进行查看，注意报表的时间跨度会有所不同。

#### B:时间选择

通过右上角时间导航栏选择时间：点击[+1h]/[-1h]切换时间为下一小时/上一小时；点击[+1d]/[-1d]切换时间为后一天的同一小时/前一天的同一小时；点击右上角[+7d]/[-7d]切换时间为后一周的同一小时/前一周的同一小时；点击[now]回到当前小时。

#### C:项目选择

输入项目名，查看项目数据；如果需要切换其他项目数据，输入项目名，回车即可。

#### D:机器分组

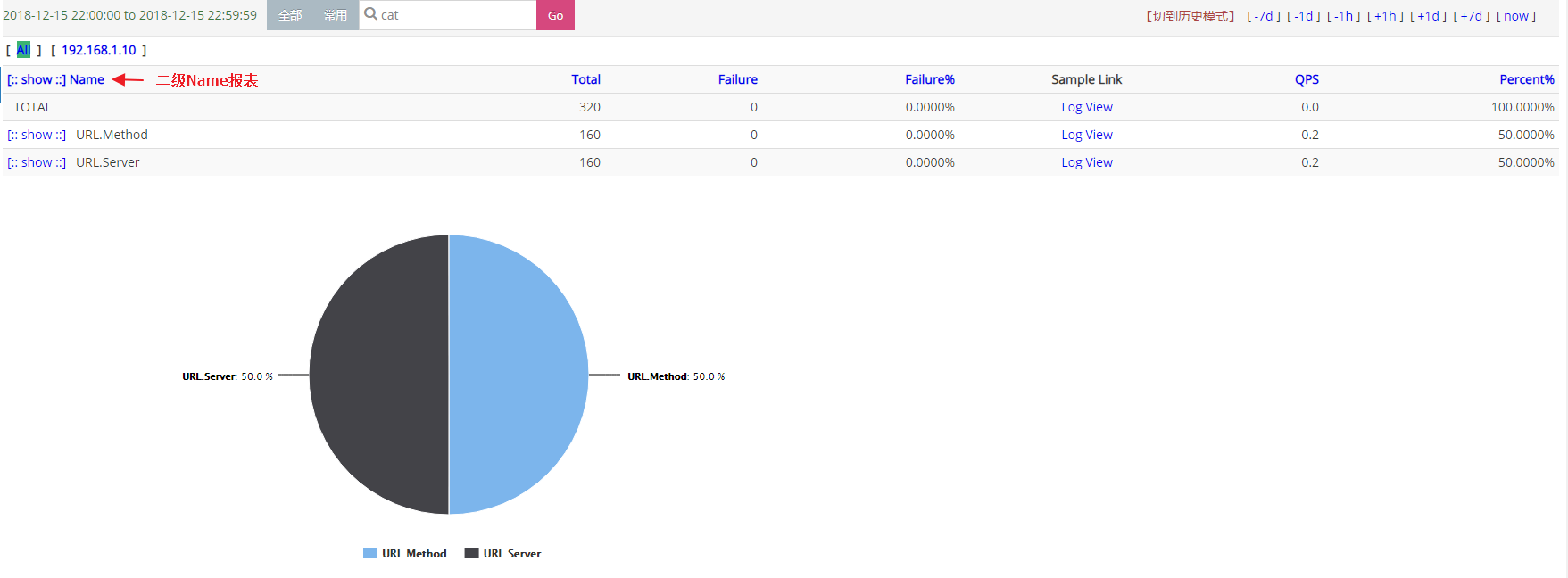
cat可以将若干个机器，作为一个分组进行数据统计。默认会有一个All分组，代表所有机器的统计数据，即集群统计数据。

#### E:所有Type汇总报表

第一层分类（Type），点击查看第二级分类（称为name）数据：

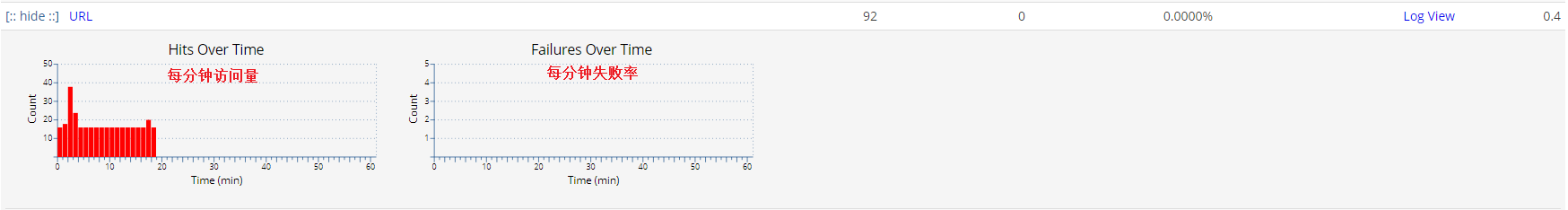
Event的埋点的Type和Name由业务自己定义，当打点了Cat.newEvent(type, name)时，第一层分类是type，第二级分类是name。

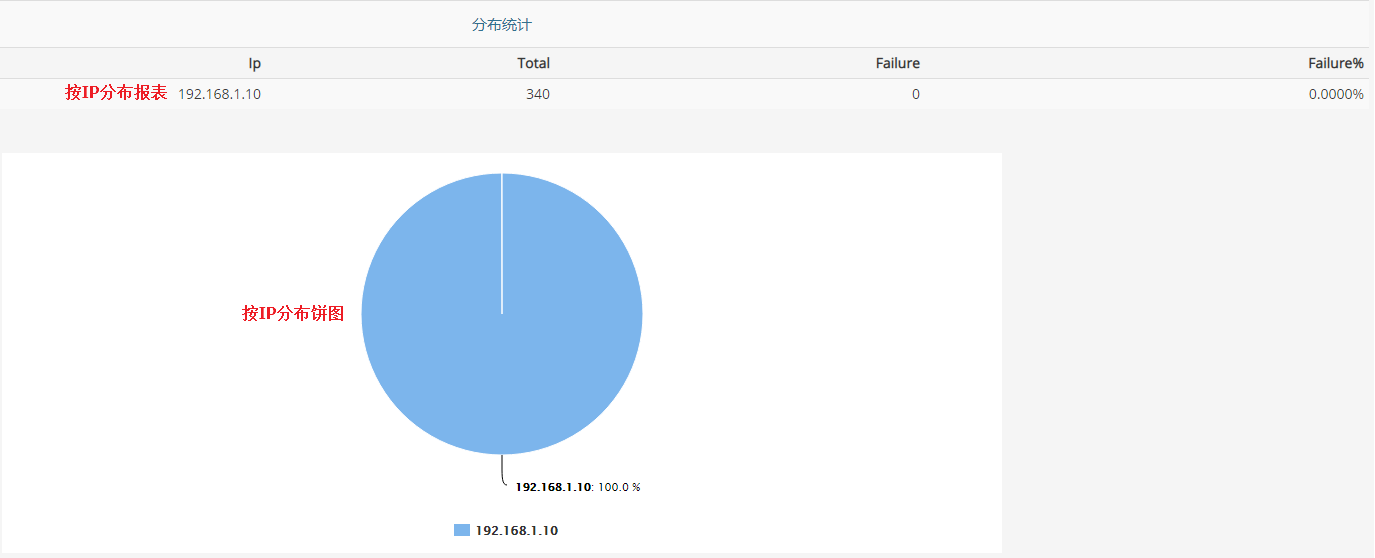
第二级分类数据叫是统计相同type下的所有name数据，如下图，数据均与第一级（type）一样的展示风格：



#### F:指标图表

显示的是小时粒度第一级分类（type）的次数、错误数、失败率等数据。





#### G:指标说明

显示的是小时粒度第一级分类（type）的次数、错误数、失败率等数据。

#### H:样本logview

sample中是采样的调用链路对应的MessageTree信息

## 3、示例说明

想记录在某个方法中一个条件分支中分支1中的函数调用了多少次，分支2中的函数调用了多少次，又失败了多少次。

以下是代码示例：

```

public void testEvent() {

for (int i = 0; i < 100; i++) {

Transaction t = Cat.newTransaction("Trans", "test");

for (int j = 0; j < 6000; j++) {

if (j % 3 == 0) {

func1();

Cat.logEvent("Func", "Func1");

} else {

boolean result = func2();

Event e = Cat.newEvent("Func", "Func2");

if (result) {

e.setSuccessStatus();

} else {

e.setStatus("False");

}

e.complete();

}

}

t.setStatus(Transaction.SUCCESS);

t.complete();

}

}

private void func1() {

}

private boolean func2() {

Random random = new Random();

int res = random.nextInt(100);

if (res % 2 == 0)

return false;

return true;

}

```

# Problem报表

## 1、主要功能

Problem记录整个项目在运行过程中出现的问题，包括一些异常、错误、访问较长的行为。Problem报表是由logview存在的特征整合而成，方便用户定位问题。 来源：

业务代码显示调用Cat.logError(e) API进行埋点，具体埋点说明可查看埋点文档。

与LOG框架集成，会捕获log日志中有异常堆栈的exception日志。

## 2、报表介绍



## 3、从上而下分析报表

### （1）、报表的时间跨度

cat默认是以一小时为统计时间跨度，点击[切到历史模式]，更改查看报表的时间跨度：默认是小时模式；切换为历史模式后，右侧快速导航，变为month(月报表)、week(周报表)、day(天报表)，可以点击进行查看，注意报表的时间跨度会有所不同。

### （2）、时间选择

通过右上角时间导航栏选择时间：点击[+1h]/[-1h]切换时间为下一小时/上一小时；点击[+1d]/[-1d]切换时间为后一天的同一小时/前一天的同一小时；点击右上角[+7d]/[-7d]切换时间为后一周的同一小时/前一周的同一小时；点击[now]回到当前小时。

### （3）、项目选择

输入项目名，查看项目数据；如果需要切换其他项目数据，输入项目名，回车即可。

### （4）、IP选择

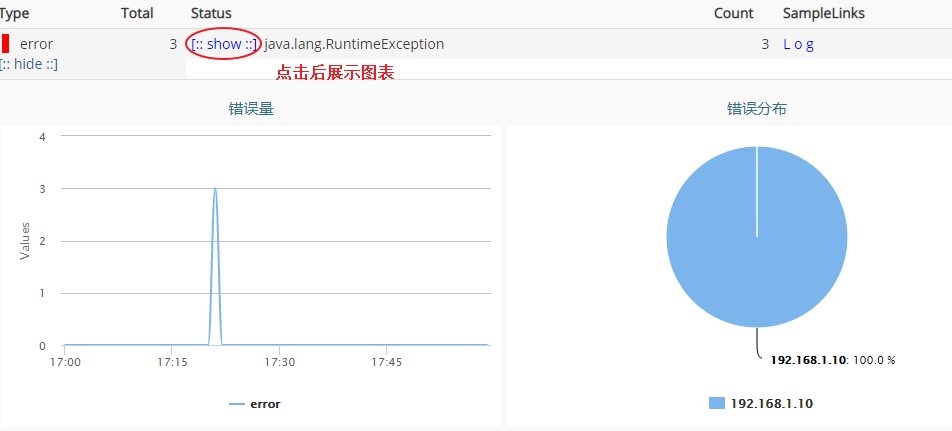
可以选择一个具体IP，默认为All，代表整个集群。

### （5）、所有错误汇总报表

第一层分类（Type），代表错误类型，比如error、long-url等；第二级分类（称为Status），对应具体的错误，比如一个异常类名等。

### （6）、错误数分布

点击type和status的show，分别展示type和status的分钟级错误数分布。

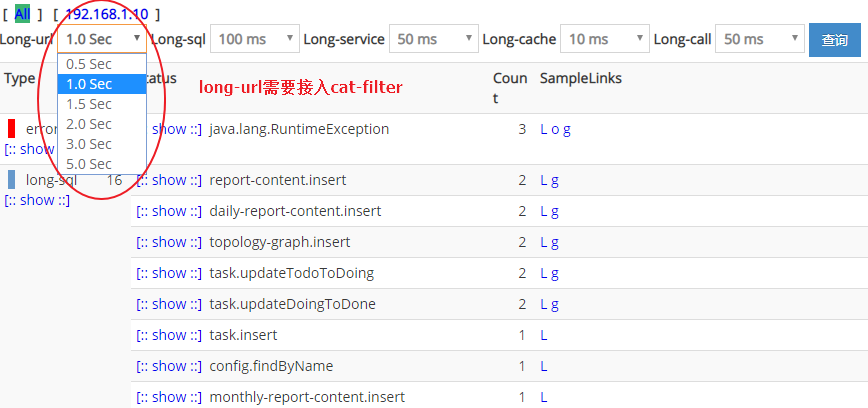


### （7）、链路sample

原始日志查看里一般会有出错的整个链路及错误堆栈，这里默认展示60条log。

### （8）、自定义阈值

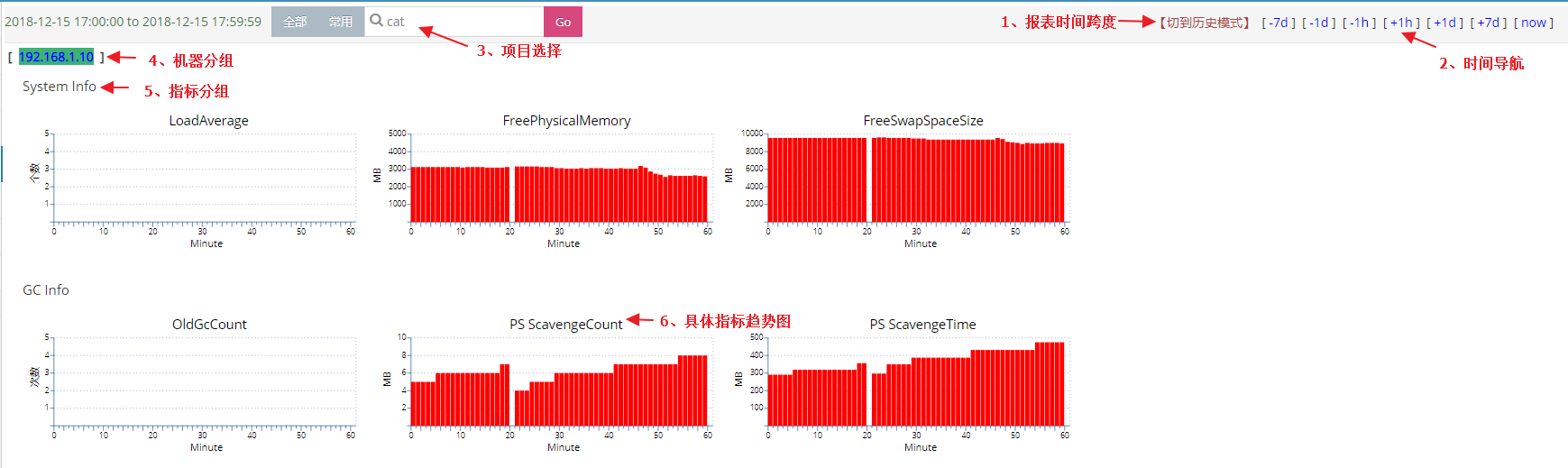
对于较长耗时，也认为是problem错误，cat可以选择不同的耗时阈值。例如：接入cat-filter后，此处可以选择long-url不同的阈值，筛选不同的长URL。



# Heartbeat报表

## 1、报表介绍

Heartbeat报表是CAT客户端，以一分钟为周期，定期向服务端汇报当前运行时候的一些状态。



## 2、从上而下分析报表

### （1）、报表时间跨度

cat默认是以一小时为统计时间跨度，点击[切到历史模式]，更改查看报表的时间跨度：默认是小时模式；切换为历史模式后，右侧快速导航，变为month(月报表)、week(周报表)、day(天报表)，可以点击进行查看，注意报表的时间跨度会有所不同。

### （2）、时间导航

通过右上角时间导航栏选择时间：点击[+1h]/[-1h]切换时间为下一小时/上一小时；点击[+1d]/[-1d]切换时间为后一天的同一小时/前一天的同一小时；点击右上角[+7d]/[-7d]切换时间为后一周的同一小时/前一周的同一小时；点击[now]回到当前小时。

### （3）、项目选择

输入项目名，查看项目数据；如果需要切换其他项目数据，输入项目名，回车即可。

### （4）、机器分组

CAT以机器为粒度展示心跳报表；选择一个具体的机器IP，查看该机器的心跳状态。

### （5）、指标分组

心跳指标分组，比如系统指标（System Info）、GC指标（GC Info）、JVM指标（JVMHeap Info）等。

### （6）、指标趋势图

以分钟为粒度，展示一个具体指标值的时间趋势。

## 3、监控指标介绍

### （1）、JVM相关指标

以下所有的指标统计都是1分钟内的值，cat最低统计粒度是一分钟。

|  |  |
| --- | --- |
| JVM GC相关指标 | 描述 |
| NewGc Count / PS Scavenge Count | 新生代GC次数 |
| NewGc Time / PS Scavenge Time | 新生代GC耗时 |
| OldGc Count | 老年代GC次数 |
| PS MarkSweepTime | 老年代GC耗时 |
| Heap Usage | Java虚拟机堆的使用情况 |
| None Heap Usage | Java虚拟机Perm的使用情况 |

|  |  |
| --- | --- |
| JVM Thread 相关指标 | 描述 |
| Active Thread | 系统当前活动线程 |
| Daemon Thread | 系统后台线程 |
| Total Started Thread | 系统总共开启线程 |
| Started Thread | 系统每分钟新启动的线程 |
| CAT Started Thread | 系统中CAT客户端启动线程 |

可以参考java.lang.management.ThreadInfo的定义

### （2）、系统指标

|  |  |
| --- | --- |
| System 相关指标 | 描述 |
| System Load Average | 系统Load详细信息 |
| Memory Free | 系统memoryFree情况 |
| FreePhysicalMemory | 物理内存剩余空间 |
| / Free | /根的使用情况 |
| /data Free | /data盘的使用情况 |

# Business报表

## 1、主要功能

Business报表对应着业务指标，比如订单指标。与Transaction、Event、Problem不同，Business更偏向于宏观上的指标，另外三者偏向于微观代码的执行情况。

场景示例：

（1）、我想监控订单数量。

（2）、我想监控订单耗时。

## 2、报表介绍



## 3、从上而下分析报表

### （1）、报表时间跨度

cat默认是以一小时为统计时间跨度，Business报表可以切换时间跨度，最大为两天。

### （2）、时间导航

通过右上角时间导航栏选择时间：点击[+1h]/[-1h]切换时间为下一小时/上一小时；点击[+1d]/[-1d]切换时间为后一天的同一小时/前一天的同一小时；点击右上角[+7d]/[-7d]切换时间为后一周的同一小时/前一周的同一小时；点击[now]回到当前小时。

### （3）、项目选择

输入项目名，查看项目数据；如果需要切换其他项目数据，输入项目名，回车即可。

### （4）、指标报表

指标报表，可以看到所有上报指标的时间趋势，有个数和平均延时两种类型指标。

## 4、基线

基线是对业务指标的预测值。

## 5、基线生成算法

最近一个月的4个每周几的数据加权求和平均计算得出，秉着更加信任新数据的原则，cat会基于历史数据做异常点的修正，会把一些明显高于以及低于平均值的点剔除。

举例：今天是2018-10-25（周四），今天整天基线数据的算法是最近四个周四（2018-10-18，2018-10-11，2018-10-04，2018-09-27）的每个分钟数据的加权求和或平均，权重值依次为1，2，3，4。如：当前时间为19:56分设为value，前四周对应的19:56分数据（由远及近）分别为A,B,C,D，则value = (A+2B+3C+4D) / 10。

对于刚上线的应用，第一天没有基线，第二天的基线基线是前一天的数据，以此类推。

## 6、如何开启基线

只有配置了基线告警的指标，才会自动计算基线。如需基线功能，请配置基线告警。

## 7、注意事项

（1）、打点尽量用纯英文，不要带一些特殊符号，例如 空格( )、分号(:)、竖线(|)、斜线(/)、逗号(,)、与号(&)、星号(\*)、左右尖括号(<>)、以及一些奇奇怪怪的字符

（2）、如果有分隔需求，建议用下划线(\_)、中划线(-)、英文点号(.)等

（3）、由于数据库不区分大小写，请尽量统一大小写，并且不要对大小写进行改动

（4）、有可能出现小数：趋势图每个点都代表一分钟的值。假设监控区间是10分钟，且10分钟内总共上报5次，趋势图中该点的值为5%10=0.5