blogbench

详细设计说明书

编 写 人：李伟钊

编写时间：2011-03-21

部 门 名：网易杭州研究院

审 核 人：

审核时间：

修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订内容简述 | 修订日期 | 版本 | 修订人 |
| 1 | 初始版本 | 2011-03-21 | 1 | 李伟钊 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 概述

### 系统目标

* 模拟真实应用，符合web应用热点分布特征

真实的模拟博客日志应用，包括表模式，表中数据的特征，操作类型，数据分布等都根据博客真实应用的统计结果设定，符合web应用热点分布特征；

* 支持多种数据库

数据库连接使用JDBC，因此可以支持多种数据库的测试，包括MySQL， PostgreSQL， Oracle，方便对不同数据库的测试结果进行对比；

* 高效

测试程序本身开销要远小于服务器的开销，在纯内存测试的情况下，测试程序的CPU开销跟服务器的开销相比不超过3：7；

* 灵活

每类操作都可单独进行测试，以方便对NTSE性能瓶颈分析与优化。

### 设计原则

* 可复用性

blogbench作为webbench的一部分，应该将一些公用组件抽象出来，方便其他的benchmark复用。

### 参考资料

Numerical Recipes, Press et al., Cambridge, 1986

## 设计要点解析

### 发表日志后新日志的访问概率

因为存在发表日志的事务，博客表会不断增长。要保证在插入新的记录之后，新插入记录的访问概率也符合Zipf分布特征，那么需要将当前博客表的所有记录的访问概率存到一个动态数组中，在发表新日志时要更新概率数组。新发表日志的访问概率计算方式是：从现有日志中按Zipf分布获得一个BlogID，其对应的概率就是新发表日志的概率。更新概率数组后需要同时更新总概率大小以及记录总数。

由于总的概率大小发生了变化（会大于1），所以下次获取的Zipf随机数的概率计算方式为：

**double** v = Math.*random*() \* sum，其中sum为所有记录的总概率。

这里需要用一个动态数组保存各个记录的概率，由于JDK实现的动态数组在数组容量不够的时候，会重新拷贝到一块更大的连续空间，为了防止出现颠簸，所以这里的动态数组需要自己实现，目前是参考NTSE中的动态数组的实现。注意在更新动态数组的时候需要做同步处理。

### UserID的两种生成方式

UserID参数有两种生成方式，一种是显示日志列表时，从所有UserID中使用zipf(user-zipf-pct%, user-zipf-res%, user-zipf-part)分布选择；另一种是从所有BlogID中使用zipf(blog-zipf-pct%, blog-zipf-res%, blog-zipf-part)分布选择BlogID后，通过匹配数组取得其对应的UserID。

UserID范围在load阶段确定，在run阶段不再增加UserID。

### CurrentTime的两种生成方式

CurrentTime也有两种生成方式，一种是直接取得当前时间，用于发表和更新日志事务；另一种是从所有BlogID中使用zipf(blog-zipf-pct%, blog-zipf-res%, blog-zipf-part)分布选择BlogID后，通过匹配数组取得其对应的publishTime，用于显示前后日志标题事务。

### 随机事务的挑选

各种事务是按比例组成的，每个测试线程拥有自己事务池，通过事务池获取下一个要执行的事务，在事务池初始化时，计算各个事务的相对比重得出事务被执行的概率。事务池根据随机概率挑选下一个要被执行的事务。

### 日志内容的生成方式

博客日志内容来源于线上库公开权限的博客记录，通过采样获得，保存为资源文件。资源文件大小在5k~6k之间，采样的日志数为10000篇。在测试程序初始化的时候，会读取资源文件，并将所有的文件内容缓存在内存中。在测试表加载数据以及执行发表日志事务的时候，生成日志内容的方式是从所有的文件中随机挑选，读取指定长度的内容。而日志摘要则是从日志内容中按照指定的摘要长度截取得到。

## 随机数算法详解

### Zipf分布随机数

Zipf分布的概率密度函数为：

P(x) = ①

由于在Zipf分布中，热点可能只是集中在某一小部分的记录中，为了让热点数据不至于太集中，所以在webbench的Zipf随机数生成器的设计中加入了一个分区的概念。即分区是等长的，所有分区的概率分布满足Zipf分布，但是分区内的记录的概率是符合均匀分布的，即相同分区中的记录概率是相等的，而不同分区的记录的概率是不同的。为达到指定的热点比例以及访问概率，需要选定合适的screw参数。下面就说明如何计算这个screw参数。

1. 如何根据热点比例以及访问概率计算扭曲系数(screw参数)

0

1

…

…

i

……

概率密度

n

上图显示了符合Zipf分布的n个分区的概率密度，其中每个矩形的高度可以根据公式计算得出:

假设要设计一种pct%的请求都集中的res%的热点数据上的Zipf分布，那么即是前

n \* pct%的分区占所有分区的面积和的res%, 即：

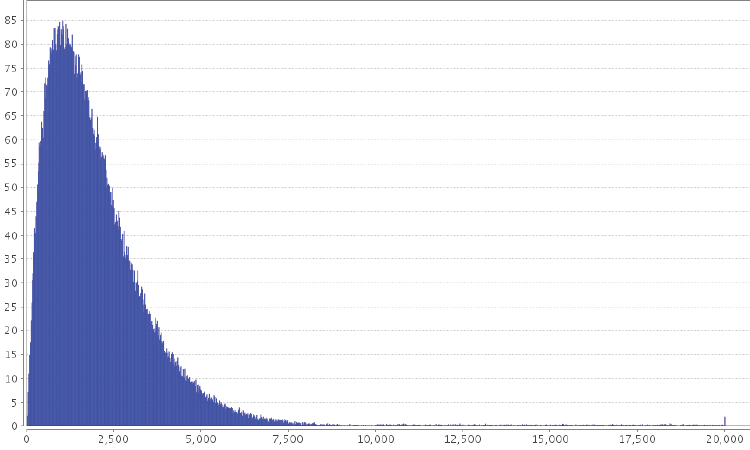
i = pct% \* n

化简可得关于screw的方程：

进一步化简可得：

这是一个关于screw的不同底的指数方程，所以可以用数值计算中的割线法或其他迭代方法求得screw一定精度内的估值。

### Gamma分布随机数



伽玛分布（Gamma distribution）是统计学的一种连续机率函数。Gamma分布中的参数α，称为形状参数（shape parameter），β称为尺度参数（scale parameter）。 如果β的值太大，那么伽玛分布将趋近于指数分布。

伽玛分布的期望和方差分别是:a/λ, a/λ~2或者α\*β, αβ~2（其中λ=β的倒数）。

Gama分布的概率密度函数为：

p(x) = , x > 0 ②

其中

, if is

, if is

, if is

这里要计算概率密度，关键在于取得α值对应的gamma函数值。参考资料第6章第1小节中提供了gamma随机数的Fortran实现及公式推导说明，通过Gamma函数的拉格朗日展开式可以推导出Gamma函数的近似值，形如：

但是实现求gamma函数的的对数值比直接求gamma函数的值简单些，所以可以将公式转化为以e为底的形式。具体的算法请参考资料。

注：也可以参考Matlab库函数中关于Gamma随机数的实现。

### 高斯分布随机数

略。

## 系统框架

### 框架图

基础构件层

统计工具

辅助工具层

随机数

通用绘图

数据库会话

逻辑层

线程管理

管理

命令行接口

接口层

事务管理

管理

操作管理

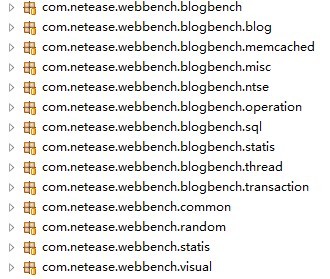
博客

SQL配置管理

参数生成器

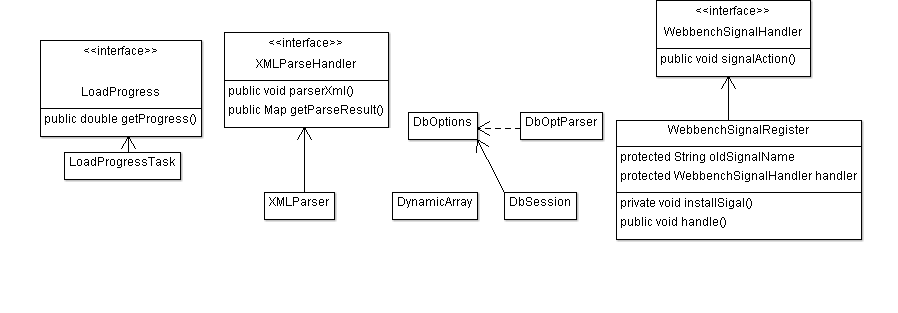
命令行选项

### 包结构



### 重要的类说明

#### 包com.netease.webbench.common



##### DynamicArray

|  |  |
| --- | --- |
| 类 |  DynamicArray |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_DynamicArray_classes.png |
| 主要属性 |  |
| 主要方法 | * public void append()   在动态数组尾部追加一个元素   * public E get()   获得指定下标的元素   * public E set()   设置指定下标的元素值 |
| 备注 | 需要保证动态数组的多线程安全。 |

##### DbOptions

|  |  |
| --- | --- |
| 类 |  DbOptions |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_DbOptions_classes.png |
| 主要属性 | 略 |
| 主要方法 | 略 |
| 备注 | 数据库选项 |

##### DbSession

|  |  |
| --- | --- |
| 类 | DbSession |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_DbSession_classes.png |
| 主要属性 | * private Connection   jdbc连接   * private DbOptions dbOpt   数据库选项 |
| 主要方法 | * public void createConnection()   创建连接   * public void createPreparedStatement()   创建预编译语句   * public ResultSet query()   需要返回结果集的查询   * public int update()   不需要返回结果集的查询 |
| 备注 | 数据库会话 |

##### DbOptParser

|  |  |
| --- | --- |
| 类 | DBOptParser |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_DbOptParser_classes.png |
| 主要属性 | 无 |
| 主要方法 | * public Pair<DbOptions, String[]> parse()   从命令行参数中解析数据库选项 |
| 备注 | 数据库选项解析器。 |

##### LoadProgress

|  |  |
| --- | --- |
| 类 | LoadProgress |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_classes.png |
| 主要属性 | 无 |
| 主要方法 | * public double getProgress()   获取进度 |
| 备注 | 获取进度接口。 |

##### LoadProgressTask

|  |  |
| --- | --- |
| 类 | LoadProgressTask |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_LoadProgressTask_classes.png |
| 主要属性 | 略 |
| 主要方法 | * public void run()   开始显示并更新进度条 |
| 备注 | 进度条显示及更新任务。 |

##### XMLParser

|  |  |
| --- | --- |
| 类 | XMLParser |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_XMLParser_classes.png |
| 主要属性 | * protected XMLParseHandler   XML解析器回调接口 |
| 主要方法 | * public void setParseHandler()   设置XML解析器回调接口   * public Map parse()   解析XML文件 |
| 备注 | XML文件解析器。 |

##### XMLParserHandler

|  |  |
| --- | --- |
| 类 | XMLParserHandler |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_XMLParserHandler_classes.png |
| 主要属性 | 无 |
| 主要方法 | * public void parseXml()   解析XML文件   * public Map getParseResult()   获得解析结果 |
| 备注 | XML文件解析回调接口。 |

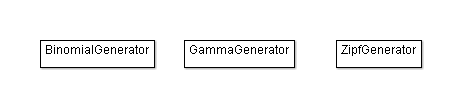
##### WebbenchSignalHandler

|  |  |
| --- | --- |
| 类 | WebbenchSignalHandler |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\WebbenchSignalHandler.png |
| 主要属性 | 无 |
| 主要方法 | * public void signalAction() |
| 备注 | 信号处理回调句柄 |

##### WebbenchSignalRegister

|  |  |
| --- | --- |
| 类 | WebbenchSignalRegister |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\WebbenchSignalRegister.png |
| 主要属性 | * protected String oldSignalName   上次收到的信号名称   * protected WebbenchSignalHandler handler   信号处理回调句柄 |
| 主要方法 | * private void installSignal()   注册要处理的信号   * public void handler()   重载sun.misc.SignalHandler的handler方法 |
| 备注 | 处理信号注册 |

#### 包com.netease.webbench.random



##### BinomialGenerator

|  |  |
| --- | --- |
| 类 | BinomialGenerator |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_BinomialGenerator_classes.png |
| 主要属性 | * private double[] probabilite   概率数组   * private int maxRecords   最大记录值   * private double perspect   期望   * private double largeUserScale   额外生成的大数的比例   * private boolean RECORD\_INCLUDE\_ZERO   返回的随机数是否可以包含0 |
| 主要方法 | * public getBinomialRandomNum()   获得服从高斯分布的随机数 |
| 备注 | 高斯分布随机数生成器。 |

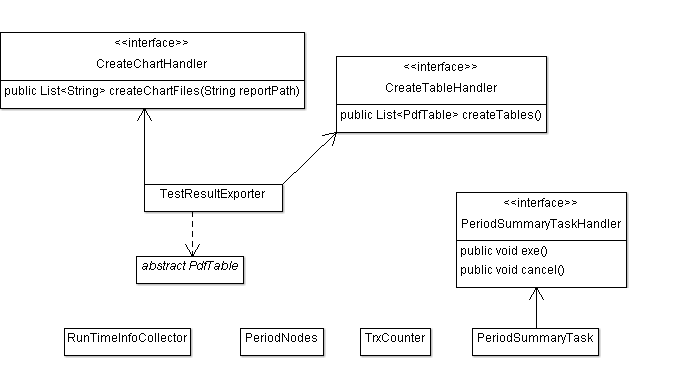
##### GammaGenerator

|  |  |
| --- | --- |
| 类 | GammaGenerator |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_GamaGenerator_classes.png |
| 主要属性 | * private double alpha   伽马分布中的alpha参数   * private double lambda   伽马分布中的lambda参数   * private double beta   伽马分布中的beta参数   * private double[] probabilities   概率数组   * private int max   返回的随机数的最大值   * private Boolean RECORD\_INCLUDE\_ZERO   返回的随机数是否包含0 |
| 主要方法 | * public getGammaRandomNum()   获得服从gamma分布的随机数   * private double getGammaPdf()   获得gamma分布的概率密度   * private double gammaln()   获得gamma函数的对数值 |
| 备注 | Gamma随机数生成器。 |

##### ZipfGenerator

|  |  |
| --- | --- |
| 类 | ZipfGenerator |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_ZipfGenerator_classes.png |
| 主要属性 | * private DynamicArray probabilities   各个记录的概率   * private double partProb   各个分区概率   * private sum   概率总和   * private double screw   扭曲系数   * private int pct   热点比例   * private int res   拥有热点的请求比例   * private int part   分区数   * private boolean recordIncludeZero   返回的随机数是否包含0   * private boolean multiThreadSafe   是否保证多线程安全 |
| 主要方法 | * public long getZipfRandomNum()   获得服从zipf分布的随机数   * public void updateProb()   更新概率数组   * private culculateScrew()   通过pct，res， part参数计算扭曲系数 |
| 备注 | Zipf随机数生成器。 |

#### 包com.netease.webbench.statis



##### CreateChartHandler

|  |  |
| --- | --- |
| 类 | CreateChartHandler |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\无标题模型_com_netease_webbench_common_CreateChartHandler_classes.png |
| 主要属性 | 无 |
| 主要方法 | * public List<String> createChartFiles()   创建统计图文件，返回统计图文件名列表 |
| 备注 | 创建统计图回调接口。测试结果导出器并不需要知道具体通过哪些类去获得统计图，只需要那些类实现此接口即可。 |

##### CreateTableHandler

|  |  |
| --- | --- |
| 类 | CreateTableHandler |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\CreateTableHandler.png |
| 主要属性 | 无 |
| 主要方法 | * public List<PdfTable> createTables()   创建PDF表格 |
| 备注 | 创建PDF表格回调接口。测试结果导出器并不需要知道具体通过哪些类去获得PDF表格，只需要那些类实现此接口即可。 |

##### TestResultExporter

|  |  |
| --- | --- |
| 类 | TestResultExporter |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\TestResultExporter.png |
| 主要属性 | * private benchmarkName   benchmark名称   * private String runSummary   汇总信息，会写入PDF文件   * private List<CreateChartHandler> createChartHandlerList   创建统计图接口列表   * private List<CreateTableHandler> createTableHandlerList   创建PDF表格接口列表 |
| 主要方法 | * public void export()   导出测试结果，生成PDF结果文件 |
| 备注 | 测试结果导出器。包括生成各种统计图表以及PDF结果文件等。 |

##### PdfTable

|  |  |
| --- | --- |
| 类 | PdfTable |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\PdfTable.png |
| 主要属性 | * protected PdfPtable   iText库的PDF表格   * protected String title   主标题   * protected String subtitle   副标题 |
| 主要方法 | 略 |
| 备注 | Pdf结果文件中的表格 |

##### PeriodSummaryTaskHandler

|  |  |
| --- | --- |
| 类 | PeriodSummaryTaskHandler |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\PeriodSummaryTaskHandler.png |
| 主要属性 | 无 |
| 主要方法 | * public void exe()   周期性任务执行   * public void cancel()   任务取消 |
| 备注 | 周期性汇总任务回调接口。周期性汇总任务的需要通过此接口去执行每个周期中需要执行的具体任务。 |

##### PeriodSummaryTask

|  |  |
| --- | --- |
| 类 | PeriodSummaryTask |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\PeriodSummaryTask.png |
| 主要属性 | * private long msInterval   执行信息汇总的周期间隔   * private PeriodSummaryTaskHandler taskHandler   周期性汇总任务回调接口 |
| 主要方法 | * public void run()   启动周期性信息汇总任务   * public void exit()   任务取消 |
| 备注 | 周期性信息汇总任务。 |

##### RunTimeInfoCollector

|  |  |
| --- | --- |
| 类 | RunTimeInfoCollector |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\RunTimeInofCollector.png |
| 主要属性 | * private Process proc   执行系统信息收集脚本的进程   * private String reportDir   结果输出目录   * private String scriptsDir   系统信息收集脚本所在的目录   * private long duration   测试持续时间 |
| 主要方法 | * public void beginCollectInfo()   开始收集系统信息   * public void stopCollectInfo()   停止收集系统信息 |
| 备注 | 系统信息收集器，只支持linux。是通过单独的进程调用系统信息收集脚本实现。 |

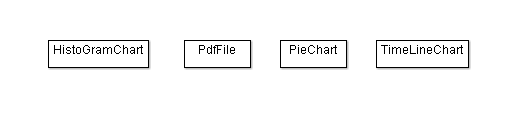
##### PeriodNodes

|  |  |
| --- | --- |
| 类 | PeriodNodes |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\PeriodNodes.png |
| 主要属性 | * private long periodCount   周期计数   * private DynamicArray<TimeNode> timeValueList   时间点统计信息列表 |
| 主要方法 | * public void addNode()   追加时间点统计信息 |
| 备注 | 周期性数值统计类，可用于统计每个周期的数值信息。 |

##### TrxCounter

|  |  |
| --- | --- |
| 类 |  TrxCounter |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\TrxCounter.png |
| 主要属性 | * private Distribution dis   分布   * private AtomicLong failedTimes   事务失败次数统计 |
| 主要方法 | * public void addTrx(long rpTime)   增加一次事务统计   * public void incrFailedTimes()   增加一次失败事务统计   * public long getMaxResponseTime()   获得最大响应时间   * public long getMinResponseTime()   获得最小响应时间   * public long getAvgResponseTime()   获得平均响应时间   * public long getMostResponseTime()   获得百分之几的响应时间   * public String getTrxName()   获得事务名称   * public String createResponseTimeChart()   生成响应时间统计图 |
| 备注 | 事务计数器，可方便地进行响应时间， 事务数等的统计。 |

#### 包com.netease.webbench.visual



##### HistoGramChart

|  |  |
| --- | --- |
| 类 | HistoGramChart |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\HistogramChart.png |
| 主要属性 | * private JFreeChart localJFreeChart;   JFreeChart对象   * private String title;   标题   * private String xLabel;   x轴标签   * private String yLabel;   y轴标签   * private String xUnit;   x轴单位 |
| 主要方法 | * public void saveToFile()   生成柱状图文件 |
| 备注 | 柱状图 |

##### PdfFile

|  |  |
| --- | --- |
| 类 | PdfFile |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\PdfFilet.png |
| 主要属性 | * private int pageHeight   PDF文件页高度   * private int pageLength   PDF文件页宽度   * private Document document   iText库PDF文档对象   * private PdfWriter writer   iText库PDF文档输出 |
| 主要方法 | * public void beginAddElement()   开始添加元素   * public void endAddElement()   结束添加元素   * public void addCover()   添加PDF文档封面   * public void addTextPage()   添加文本页   * public void addAllImageFile()   将所有图片添加为图片页   * public void addImageFile()   将图片添加为图片页   * public void addTablePage()   添加表格页 |
| 备注 | PDF文件 |

##### PieChart

|  |  |
| --- | --- |
| 类 | PieChart |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\PieChart.png |
| 主要属性 | * private JFreeChart localJFreeChart;   JFreeChart对象   * private DefaultPieDataset localDefaultPieDataset;   饼图数据集   * private String title;   标题 |
| 主要方法 | * public void addPie()   增加饼图   * public void createChart()   创建JFreeChart   * public void saveToFile()   将饼图保存为图片文件 |
| 备注 | 饼图 |

##### TimeLineChart

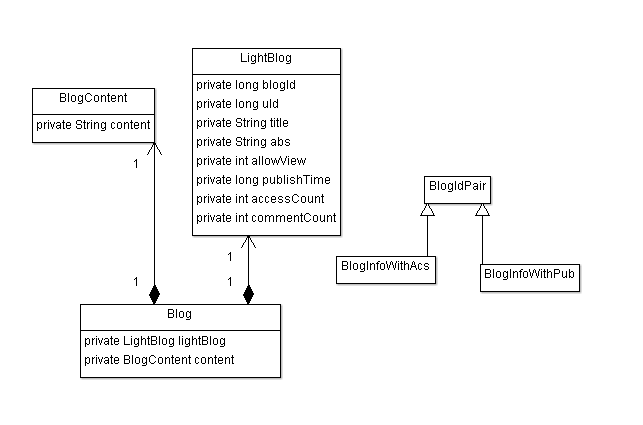
|  |  |
| --- | --- |
| 类 | TimeLineChart |
| 类图 | C:\Users\Ricky\Desktop\TimeLineChart_副本.png |
| 主要属性 | * private TimeSeries[] tsArr;   时序数据集   * private String title;   曲线图主标题   * private JFreeChart chart;   JFreeChart对象   * private int lineCount;   曲线图中的曲线计数   * private int lineMaxCount;   可添加的最大曲线数 |
| 主要方法 | * public int newline()   添加新的曲线，返回曲线编号   * public void addPoint()   在指定的曲线中添加一个数据点   * public void createChart()   创建JFreeChart   * public void saveToFile()   将时序图保存为图片文件   * public void setStartTime()   手工设置时序图开始时间   * public void setEndTime()   手工设置时序图终止时间 |
| 备注 | 时序图 |

#### 包com.netease.webbench.blogbench

##### BlogbenchTest

|  |  |
| --- | --- |
| 类 | BlogbenchTest |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BlogbenchTest.png |
| 主要属性 | 略 |
| 主要方法 | * private void checkDflOptions()   检查默认参数   * public void init()   blogbench初始化   * public void runTest()   开始blogbench测试   * protected void checkServerCharaSet()   检查服务器字符集设置   * protected void checkServerIsAlive()   判断服务器是否可用 |
| 备注 | Blogbench测试程序类，单实例 |

#### 包com.netease.webbench.blogbench.blog



##### Blog

|  |  |
| --- | --- |
| 类 | Blog |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\Blog.png |
| 主要属性 | * private LightBlog lightBlog   除日志内容之外的其他信息   * private BlogContent content   博客日志内容 |
| 主要方法 | 略 |
| 备注 | 博客类 |

##### LightBlog

|  |  |
| --- | --- |
| 类 | LightBlog |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\lightBlog.png |
| 主要属性 | 略 |
| 主要方法 | 略 |
| 备注 | 轻量博客类，不包含日志内容。 |

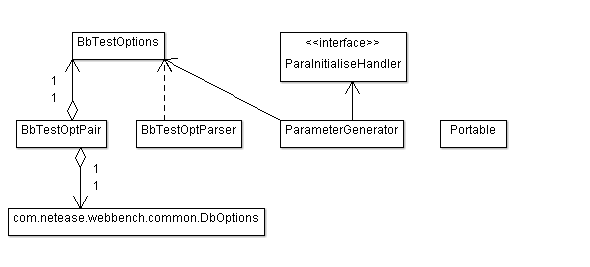
##### BlogContent

|  |  |
| --- | --- |
| 类 | BlogContent |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BlogContent.png |
| 主要属性 | 略 |
| 主要方法 | 略 |
| 备注 | 博客日志内容类 |

##### BlogIdPair

|  |  |
| --- | --- |
| 类 | BlogIdPair |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\blogIdPair.png |
| 主要属性 | 略 |
| 主要方法 | 略 |
| 备注 | 博客ID以及用户ID组合 |

#### 包com.netease.webbench.blogbench.misc



##### BbTestOptions

|  |  |
| --- | --- |
| 类 | BbTestOptions |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestOptions.png |
| 主要属性 | 略 |
| 主要方法 | 略 |
| 备注 | blogbench测试选项 |

##### BbTestOptParser

|  |  |
| --- | --- |
| 类 | BbTestOptParser |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestOptParser.png |
| 主要属性 | 无 |
| 主要方法 | * static public Pair<BbTestOptions, String[]> parse()   解析blogbench测试选项   * static public boolean parseBoolean()   解析布尔值 |
| 备注 | blogbench测试选项解析器 |

##### BbTestOptPair

|  |  |
| --- | --- |
| 类 | BbTestOptPair |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestOptPair.png |
| 主要属性 | 略 |
| 主要方法 | 略 |
| 备注 | blogbench测试选项及数据库选项组合 |

##### ParameterGenerator

|  |  |
| --- | --- |
| 类 | ParameterGenerator |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\ParameterGenerator.png |
| 主要属性 | * private ZipfGenerator blgIndexGenerator;   博客记录索引生成器   * private ZipfGenerator userIdGenerator;   用户ID生成器   * private GammaGenerator contentLenGenerator;   博客日志长度生成器   * private DynamicArray<BlogInfoWithPub> blgArr   存储所有博客对应关系的数组   * private BbTestOptions bbTestOpt;   blogbench测试选项   * private AtomicLong maxBlogId;   当前最大博客ID   * private ArrayList<byte []> cntInitArr;   博客内容资源缓存   * private long tableRunTimeSize;   当前测试表记录数   * private ParaInitialiseHandler initialiseHandler;   参数生成器初始化回调句柄 |
| 主要方法 |  |
| 备注 | blogbench测试参数生成器 |

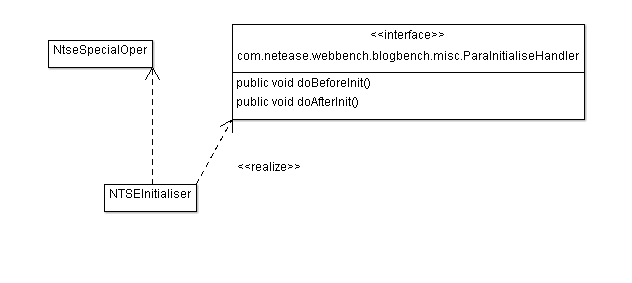
##### ParameterInitialiseHandler

|  |  |
| --- | --- |
| 类 | ParameterInitialiseHandler |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\ParameterInitialiseHandler.png |
| 主要属性 | 无 |
| 主要方法 | * public void doBeforeInit()   参数生成器初始化前执行   * public void doAfterInit()   参数生成器初始化后执行 |
| 备注 | 参数生成器初始化回调句柄。需要在参数生成器初始化前及初始化后干额外工作的，只需要实现此接口即可。例如NTSE在执行全表查询时关闭MMS。 |

##### Portable

|  |  |
| --- | --- |
| 类 | Portable |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\Portable.png |
| 主要属性 | * public static String characterSet   字符集设置 |
| 主要方法 | * static public void setCharacterSet()   设置字符集   * static public String getCharacterSet()   获得字符集设置   * static public String getDflJdbcDrvName()、   获得默认的JDBC driver名称   * static public String getDflJdbcUrl()   获得默认的JDBC URL   * static public boolean surportMultiInsert()   数据库是否支持multi-insert   * getBlogContentTableName()   当使用两张测试表时，content表的表名 |
| 备注 | 可移植信息相关 |

#### 包com.netease.webbench.blogbench.ntse



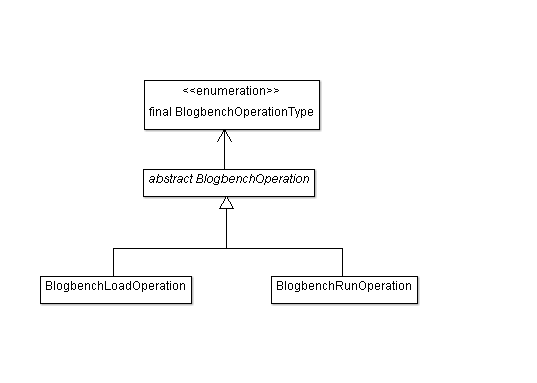
##### NtseSpecialOper

|  |  |
| --- | --- |
| 类 | NtseSpecialOper |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\NtseASpecialOper.png |
| 主要属性 | 无 |
| 主要方法 | * static public String setNtseIndexBuildAlgorithm()   设置NTSE索引创建算法   * static public void disableMms()   NTSE存储引擎中禁止使用MMS   * static public void enableMms()   NTSE存储引擎中恢复使用MMS |
| 备注 | NTSE存储引擎相关的特殊操作 |

##### NTSEInitialiser

|  |  |
| --- | --- |
| 类 | NTSEInitialiser |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\NTSEInitialiser.png |
| 主要属性 | 略 |
| 主要方法 | 略 |
| 备注 | 参考接口ParameterInitialiseHandler |

#### 包com.netease.webbench.blogbench.operation



##### BlogbenchOperationType

|  |  |
| --- | --- |
| 类 | BlogbenchOperationType |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BlogbenchOperationType.png |
| 主要属性 | * public static final LOAD   LOAD操作   * public static final RUN   RUN操作 |
| 主要方法 | 无 |
| 备注 | blogbench操作类型(LOAD/RUN) |

##### BlogbenchOperation

|  |  |
| --- | --- |
| 类 | BlogbenchOperation |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BlogbenchOperation.png |
| 主要属性 | * private DbOptions dbOpt   数据库选项   * protected BbTestOptions bbTestOpt   blogbench测试选项   * protected ParameterGenerator paragon   参数生成器   * protected BlogbenchOperationType operType、   操作类型 |
| 主要方法 | * abstract public void executeOper()   执行操作   * public void signalAction()   见WebbenchSignalHandler接口 |
| 备注 | blogbench操作 |

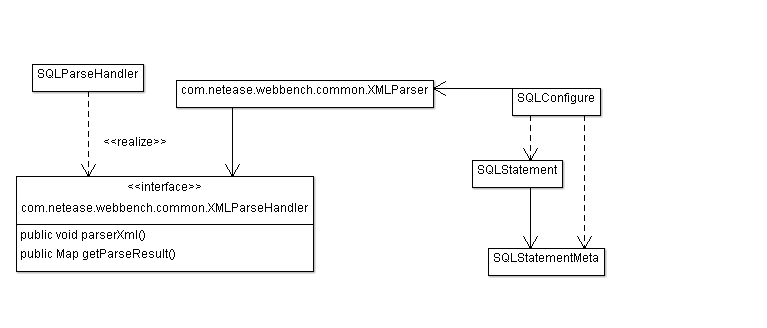
##### BlogbenchLoadOperation

|  |  |
| --- | --- |
| 类 | BlogbenchLoadOperation |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BlogbenchLoadOperation.png |
| 主要属性 | * private BbTestInsertThread[] insertThrdGrp   插入线程组   * private DbSession dbSession   数据库会话 |
| 主要方法 | * public void executeOper()   执行操作   * public void load()   加载数据   * private boolean createThrdAndLoadData()   创建插入线程组并加载数据   * private String getLoadSummary()   获得加载数据统计汇总信息 |
| 备注 | blogbench LOAD操作 |

##### BlogbenchRunOperation

|  |  |
| --- | --- |
| 类 | BlogbenchRunOperation |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BlogbenchRunOperation.png |
| 主要属性 | * private BlogbenchCounters blogbenchCounters   事务计数器   * private BbPeriodSummaryTaskHandler periodTaskHandler   周期汇总任务回调接口   * private PeriodSummaryTask periodSummaryTask   周期汇总任务   * private ThreadRunFlagTimer runFlagTimer   线程运行标记计时器   * private BbTestRunThread[] trdArr   工作线程组   * private RunTimeInfoCollector runTimeInfoCollector   系统信息收集器   * private trxTypeNum   事务类型数 |
| 主要方法 | * public void executeOper()   执行操作   * public String getSummary()   获得汇总信息   * private void finish()   结束测试   * private void exportResult()   导出测试结果   * private void initMemcached()   初始化与Memcached连接   * private long getActualTestTime()   获得实际执行时间   * private void printResult()   打印测试结果到标准输出   * private String getTrxResult()   获得事务结果统计汇总信息 |
| 备注 | blogbench RUN操作 |

#### 包com.netease.webbench.blogbench.sql



##### SQLStatementMeta

|  |  |
| --- | --- |
| 类 | SQLStatementMeta |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\SQLStatementMeta.png |
| 主要属性 | * private String name   语句名   * private boolean useMemcached   是否使用memcached   * private boolean useTwoTables   是否使用两张测试表 |
| 主要方法 | 略 |
| 备注 | SQL语句元信息，包括语句名，是否使用memcached，是否是两张测试表时的语句。 |

##### SQLStatement

|  |  |
| --- | --- |
| 类 | SQLStatement |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\SQLStatement.png |
| 主要属性 | * private SQLStatementMeta stmtMeta   SQL语句元信息   * private String sqlStmt   SQL语句 |
| 主要方法 | 略 |
| 备注 | SQL语句 |

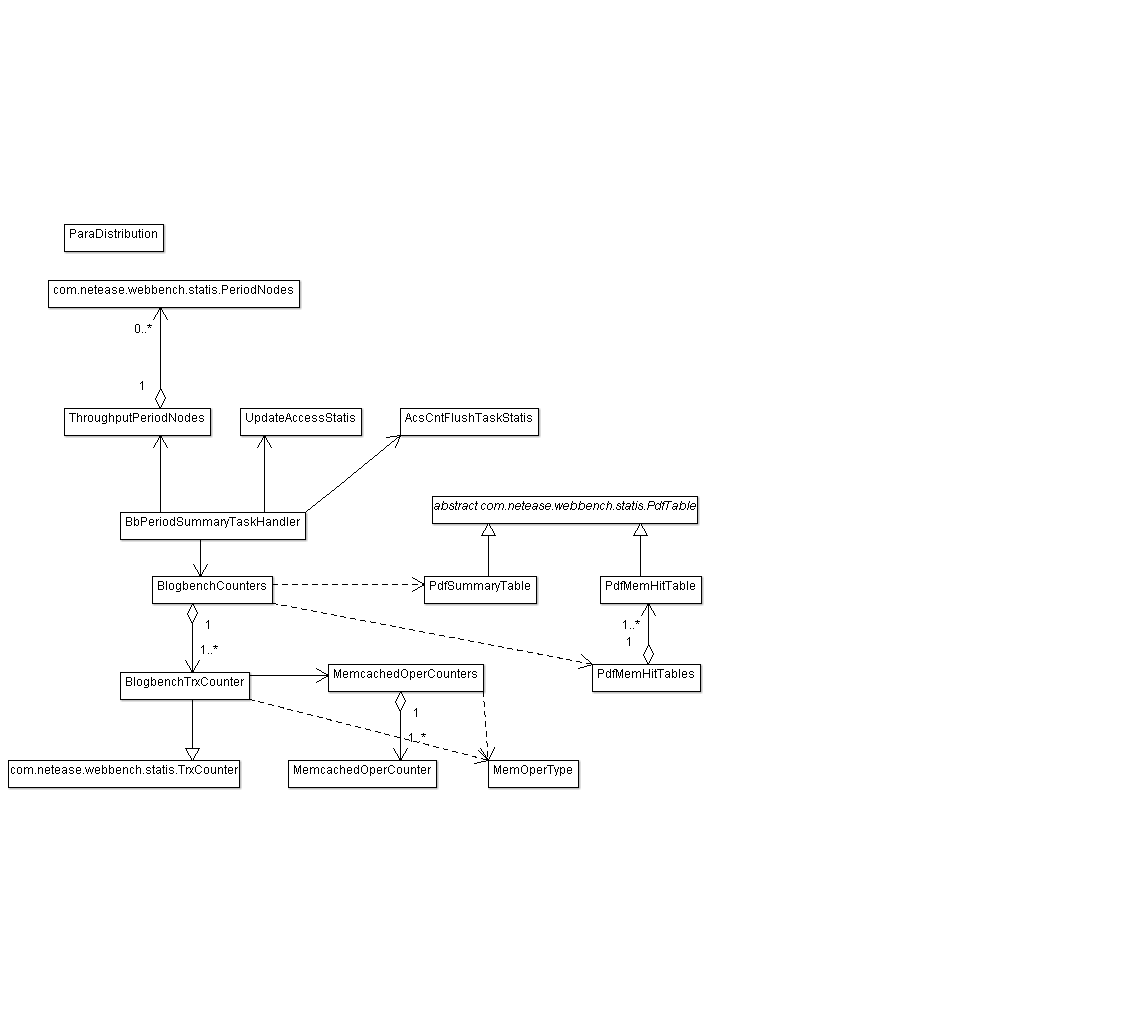
##### SQLParseHandler

|  |  |
| --- | --- |
| 类 | SQLParseHandler |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\SQLParseHandler.png |
| 主要属性 | * private String filename   配置文件路径   * private Map<String, Map<SQLStatementMeta, SQLStatement>> stmtMaps   配置文件解析结果 |
| 主要方法 | * public void parseXml()   见接口XMLParseHandler   * public Map<…> getParseResult()   见接口XMLParseHandler |
| 备注 | SQL语句配置信息解析。 |

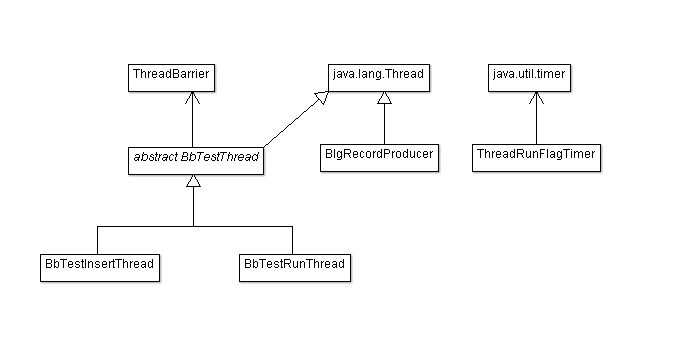
##### SQLConfigure

|  |  |
| --- | --- |
| 类 | SQLConfigure |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\SQLConfigure.png |
| 主要属性 | * protected String dbType   数据库类型   * protected XMLParser parser   XML文件解析器   * protected Map<…> stmtMapts   SQL语句配置解析结果 |
| 主要方法 | 略 |
| 备注 | SQL语句配置，单实例。 |

#### 包com.netease.webbench.blogbench.statis



#### 包com.netease.webbench.blogbench.thread



##### ThreadBarrier

|  |  |
| --- | --- |
| 类 | ThreadBarrier |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\ThreadBarrier.png |
| 主要属性 | 略 |
| 主要方法 | * public void waitBarrier()   阻塞等待线程障碍移除   * public void removeBarrier()   移除线程障碍 |
| 备注 | 线程障碍，用于同步所有的线程进入等待状态。 |

##### ThreadRunFlagTimer

|  |  |
| --- | --- |
| 类 | ThreadRunFlagTimer |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\ThreadRunFlagTimer.png |
| 主要属性 | * private boolean runFlag   继续运行标记   * private boolean isTimeout   是否超时 |
| 主要方法 | * public void start()   开始计时   * public boolean getRunFlag()   获得线程运行标记   * public boolean setExpired()   手工设置线程运行标志为False |
| 备注 | 线程执行标记计数器。用于控制测试的时间， 当设定的测试时间超时时，线程执行标记将被改为False，而测试工作线程每次执行完一次事务都会检查这个标记，当为False的时候， 工作线程将退出。 |

##### BbTestThread

|  |  |
| --- | --- |
| 类 | BbTestThread |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestThread.png |
| 主要属性 | * protected DbSession dbSession   数据库会话   * protected ParameterGenerator paraGen   参数生成器   * protected ThreadBarrier barrier   线程障碍   * protected boolean isWaiting   线程是否被阻塞   * protected int exitErrCode   测试线程退出码，非0即为测试过程发生错误 |
| 主要方法 | * protected void myWait()   阻塞进入同步状态(线程障碍被移除时被唤醒)   * public int getErrorCode()   获得测试线程退出错误码 |
| 备注 | blogbench线程抽象类 |

##### BbTestInsertThread

|  |  |
| --- | --- |
| 类 | BbTestInsertThread |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestInsertThread.png |
| 主要属性 | * private BlgRecordProducer blogRcdProducer   博客记录生产者   * private long rcdCntToInsert   本线程要插入的记录数   * private long recordInserted   已经插入的记录数 |
| 主要方法 | * protected void doLoadData()   加载数据   * protected void doMultiInsert()   执行multi-insert   * protected void doInsertOneTable()   只使用一张测试表时的插入   * protected void doInsertTwoTable()   使用两张测试表时的插入 |
| 备注 | blogbench LOAD操作时执行插入的工作线程。 |

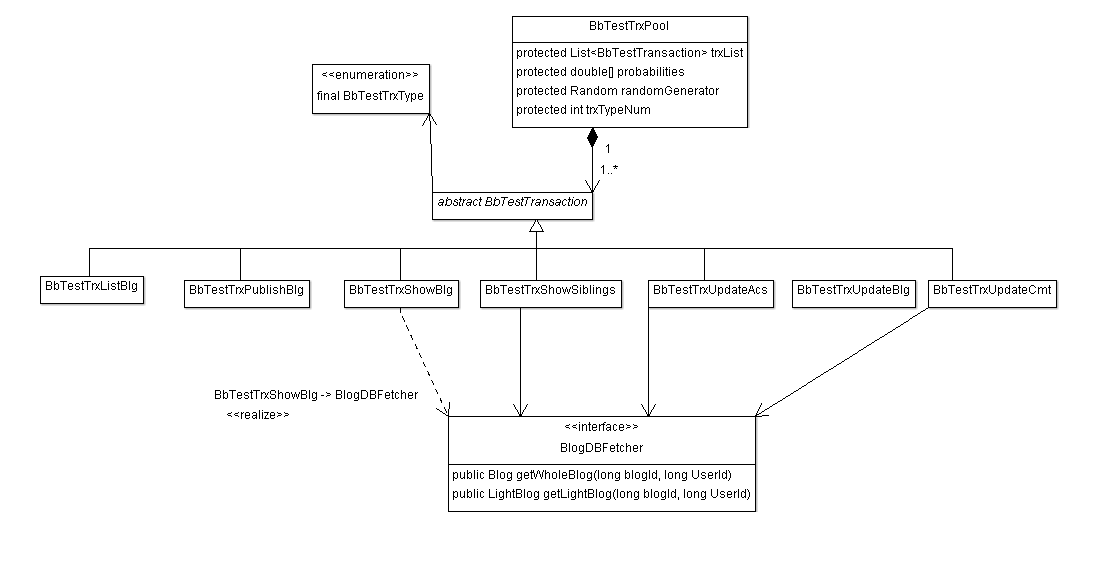
##### BbTestRunThread

|  |  |
| --- | --- |
| 类 | BbTestRunThread |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestRunThread.png |
| 主要属性 | * private BbTestTrxPool trxPool   事务池   * private BlogbenchTYrxCounter totalTrxCounter   所有事务的总计数器   * private ThreadRunFlagTimer runFlagTimer   线程运行标志计时器 |
| 主要方法 | 略 |
| 备注 | blogbench RUN操作时执行各种事务的工作线程。 |

##### BlogRecordProducer

|  |  |
| --- | --- |
| 类 | BlogRecordProducer |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BlogRecordProducer.png |
| 主要属性 | * private BlockingQueue<Blog> blockingQueue   生成的博客记录放入此队列   * private int queueMaxSize   最大队列长度   * private long produceNum   需要生成的博客记录数   * private long hasProduced   已经生成的博客记录数   * private boolean shouldExit   是否被设置为强制退出 |
| 主要方法 | * public void forceExit()   强制线程退出   * public Blog getBlog()   获得生成的博客记录 |
| 备注 | blogbench LOAD操作时用于异步生成博客记录的生产者，插入线程将充当消费者。 |

#### 包com.netease.webbench.blogbench.transaction



##### BbTestTrxType

|  |  |
| --- | --- |
| 类 | BbTestTrxType |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestTrxType.png |
| 主要属性 | * protected String name   事务名称 |
| 主要方法 | * static public int getTrxIndex()   获得事务索引值   * static public String getTrxName()   获得事务名称 |
| 备注 | blogbench 事务类型 |

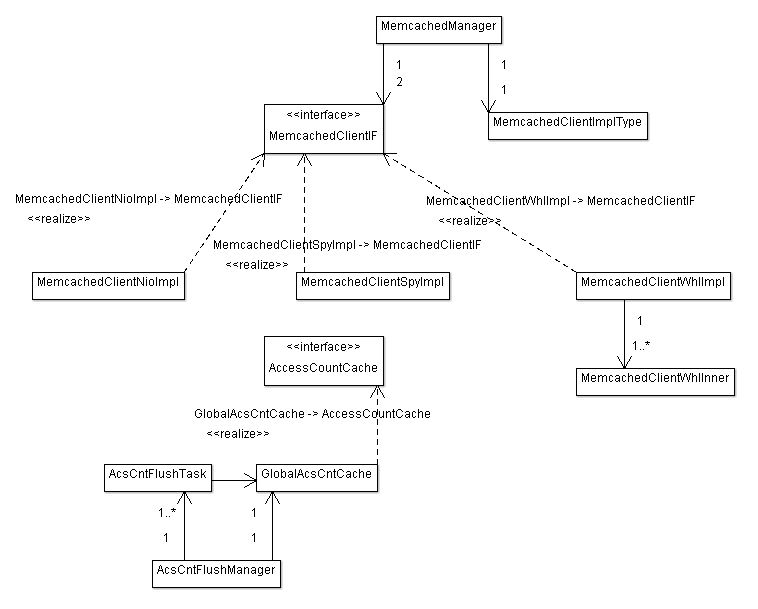
##### BbTestTransaction

|  |  |
| --- | --- |
| 类 | BbTestTransaction |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestTransaction.png |
| 主要属性 | * protected BlogbenchTrxCounter totalTrxCounter   所有事务总的计数器   * protected int pct   事务比例   * protected BbTestTrxType trxType   事务类型   * protected DbSession dbSession   数据库会话   * protected BbTestOptions bbTestOpt   blogbench测试选项 |
| 主要方法 | * public void prepare()   生成预编译语句   * public void exeTrx()   执行事务 |
| 备注 | blogbench事务抽象类 |

##### BbTestTrxPool

|  |  |
| --- | --- |
| 类 | BbTestTrxPool |
| 类图 | C:\Users\Ricky\Desktop\blogbench类图\class_single\BbTestTrxPool.png |
| 主要属性 | * protected List<BbTestTransaction> trxList   事务列表   * protected double[] probabilities   各个事务的概率   * protected int trxTrxNum   事务类型数 |
| 主要方法 | * private void initialiseTrxs()   初始化事务池   * private void initialiseProb()   初始化概率数组   * public BbTestTransactions getRandomTrx()   随机获得一个事务 |
| 备注 | blogbench事务池。事务池会根据指定的比例初始化各个事务被执行的概率，以后工作线程只需要从事务池中获取事务去执行即可。每个测试工作线程都有一个事务池。 |

#### 包com.netease.webbench.blogbench.memcached



## 测试结果可视化

### dbt-2测试结果可视化方式

1. 首先，不同线程执行测试的响应时间都写到一个叫mix.log的测试结果文件中，其中包含了每一次查询的事务类型，响应时间，线程id等信息，不同线程在对mix.log执行写操作前需要先加锁实现同步；
2. 然后，使用perl脚本解析mix.log文件，统计各类事务的响应时间最大值，最小值，平均值，90%响应时间，perl脚本通过调用gnuplot命令把统计的信息打印成图片，并生成一个index.html文件，嵌入生成的图片；
3. 系统信息的收集：在测试开始前需要手动启动命令vmstat以及iostat，命令输出信息保存成文本文件，测试结束终止命令，然后使用dbt-2下提供的shell脚本和perl脚本解析输出信息文件，并分别把各个参数的变化信息保存成图片，嵌入index.html；
4. 通过sar也可以进行系统信息的收集，开源的ksar程序可以对sar的结果进行解析，最后可以保存为图片或PDF这个程序独立于dbt-2的脚本，使用比较方便。ksar采用JFreeChart库进行绘图。

### blogbench测试结果的可视化

1. 通过对比两种可视化方式，使用JFreeChart较方便。blogbench参考ksar，使用JFreeChart来绘图，以后的扩展性也比较强，可以绘制其他需要的统计信息，而不仅仅局限于dbt-2的脚本。 另一方面，各种事务的响应时间并不用保存为文本再进行解析，再去分析dbt-2 的脚本如何解析各种事务的测试结果只会增加实现的难度；
2. 通过ksar的源码知道其保存为PDF使用iText库，所以最后blogbench的多张图片也可以归结为一个PDF文件；
3. 操作系统信息的收集将采用dbt-2的收集脚本，不重新实现；