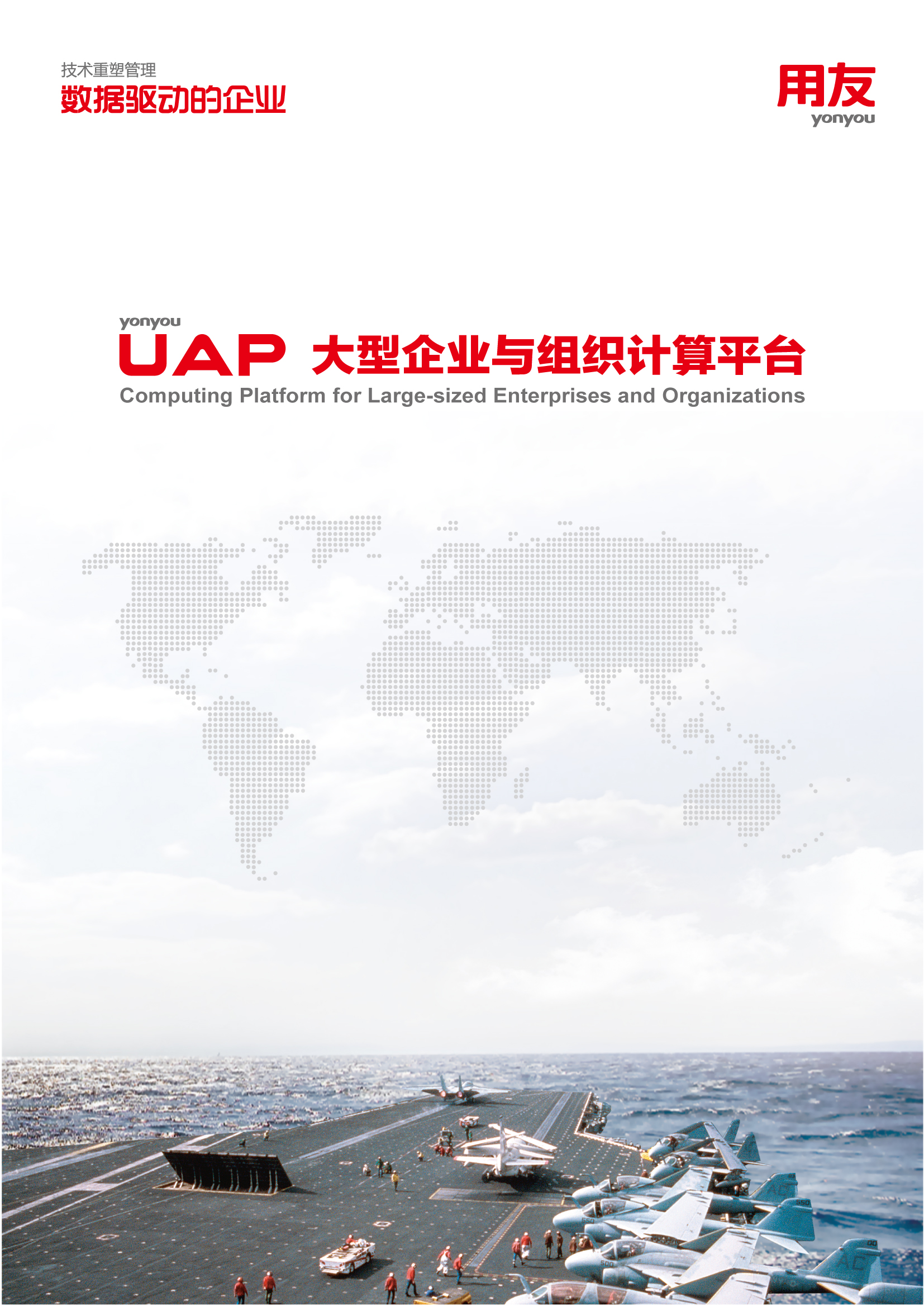
****

**数据平台性能基准测试报告**

**Version 2.1**

**OLAP引擎**

# 版权

©2014用友集团版权所有。

未经用友集团的书面许可，本发版说明任何整体或部分的内容不得被复制、复印、翻译或缩减以用于任何目的。本发版说明的内容在未经通知的情形下可能会发生改变，敬请留意。请注意：本发版说明的内容并不代表用友软件所做的承诺。

# 目录

[版权 2](#_Toc402461863)

[目录 3](#_Toc402461864)

[1 基准环境 4](#_Toc402461865)

[1.1 硬件环境 4](#_Toc402461866)

[1.2 拓扑结构 4](#_Toc402461867)

[1.3 软件环境 5](#_Toc402461868)

[2 测试场景 6](#_Toc402461869)

[2.1 测试数据 6](#_Toc402461870)

[2.2 测试模型 6](#_Toc402461871)

[2.3 环境参数 8](#_Toc402461872)

[2.4 基准指标 8](#_Toc402461873)

[3 数据 9](#_Toc402461874)

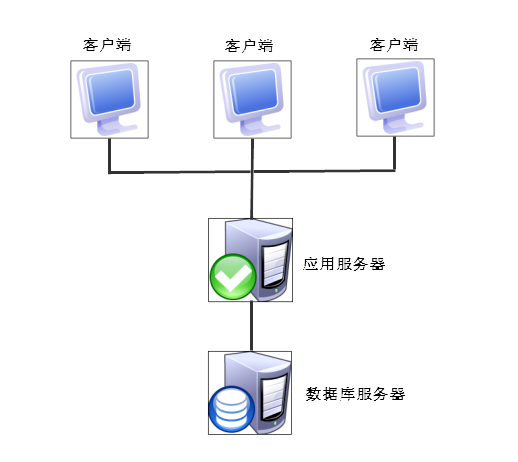
[4 附录 12](#_Toc402461875)

# 基准环境

## 硬件环境

|  |  |  |
| --- | --- | --- |
|  | **应用服务器（虚拟机）** | **数据库服务器（虚拟机）** |
| IP | 20.10.80.15 | 20.10.80.3 |
| 操作系统 | Microsoft Windows 2003 Whistler Server 32-bit (5.2) | Microsoft Windows 2003 Whistler Server 32-bit (5.2) |
| CPU | Intel QEMU Virtual CPU version (cpu64-rhel6) (1995 Mhz, 8 Socket, 1 Cores Per Socket) | Intel QEMU Virtual CPU version (cpu64-rhel6) (1995 Mhz, 8 Socket, 1 Cores Per Socket) |
| 内存 | 32G(其中JDK最大内存1G) | 32G |
| 硬盘带宽 | 100G | 100G |
| 网络 | 100Mbps | 100Mbps |

## 拓扑结构



## 软件环境

操作系统，数据库，中间件，jdk，uap runner版本等依赖的软件环境

|  |  |
| --- | --- |
| **名称** | **值** |
| 应用服务器操作系统 | Microsoft Windows 2003 Whistler Server 32-bit (5.2) |
| 数据库服务器 | Microsoft Windows 2003 Whistler Server 32-bit (5.2) |
| JDK版本 | 1.6 |
| UAPRUNNER版本 | 2.1.0 |
| NMC版本 | 6.5.0（在UAP home根目录下） |
| 环境地址 | http://20.10.80.15:80 |
| 登录用户 | dev1/ae1234 |
| design数据源信息 | ae\_design/1@20.10.130.235:1521/ORCL |
| dw数据源信息 | pi\_dw\_1g/1@20.10.80.3:1521/aepidw |

# 测试场景

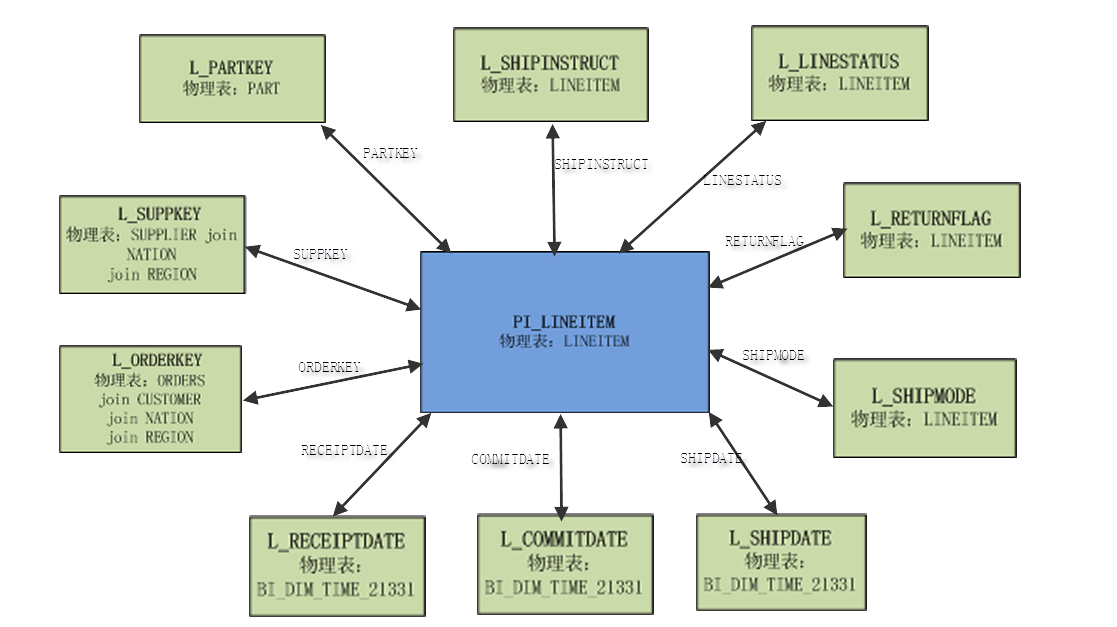
## 测试数据

TPCH 1G数据

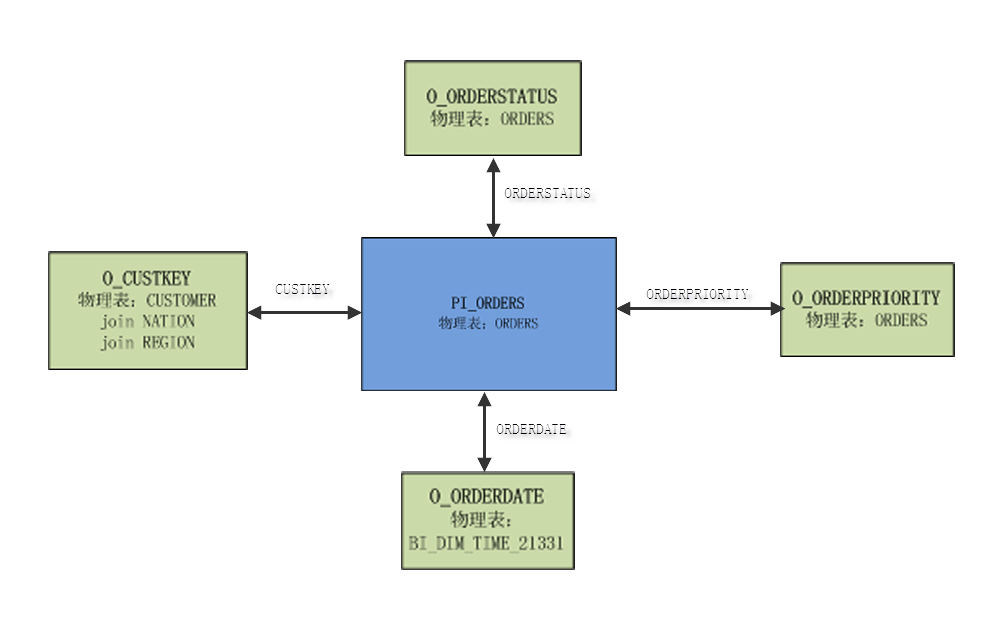
|  |  |  |
| --- | --- | --- |
| **表名** | **记录数** | **复杂度情况** |
| LINEITEM | 6,001,215 | 复杂 |
| CUSTOMER | 150,000 | 中等 |
| SUPPLIER | 10,000 | 中等 |
| PART | 200,000 | 中等 |
| PARTSUPP | 800,000 | 中等 |
| ORDERS | 1,500,000 | 复杂 |
| REGION | 5 | 简单 |
| NATION | 25 | 简单 |
| BI\_DIM\_TIME\_21331 | 8,308 | 简单 |

## 测试模型

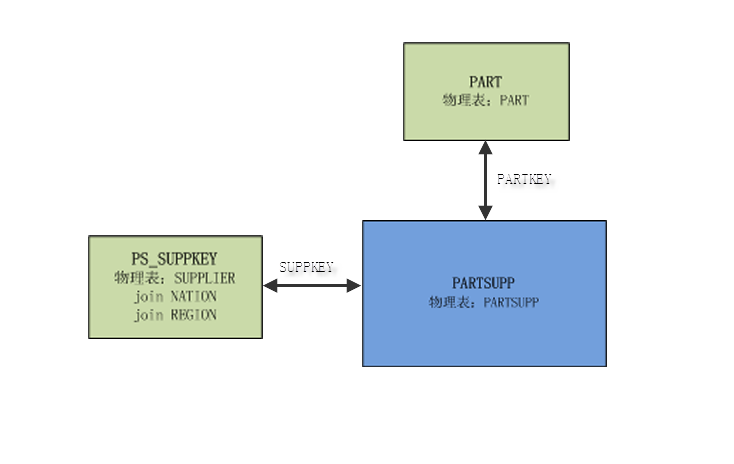
分析主题：items



分析主题：orders



分析主题：partsupp



## 环境参数

**单人执行方案：**

每个测试用例执行100次，记录用例执行时间和对应的数据库执行时间。

在单独执行和并发执行前，先执行一次查询，保证函数定义等加载到内存中。

**并发执行方案：**

  总用户数：18 人

用户上线策略：整体，每隔 10 秒上线 1 人

  用户执行策略：按时间（分钟/整体），执行 60分钟



## 基准指标

**单人执行方案：**

用例执行时间的最大值、最小值、平均值。

对应的数据库执行时间的最大值、最小值、平均值。

**并发执行方案：**

|  |  |
| --- | --- |
| **指标名称** | **说明** |
| 成功事务总数 | 通过UAPRUNNER |
| 失败事务总数 | 通过UAPRUNNER |
| 每秒通过事务数 | 通过UAPRUNNER，最小值、最大值、平均值 |
| 事务平均响应时间 | 通过UAPRUNNER，最小值、最大值、平均值 |
| 资源利用率 | 包括CPU利用率，内存利用率 |
| 数据库连接次数最多的远程调用 | 数据库连接数和执行SQL数可不对外发布，作为每一版的性能评定标准 |
| SQL数量最多的远程调用 |  |

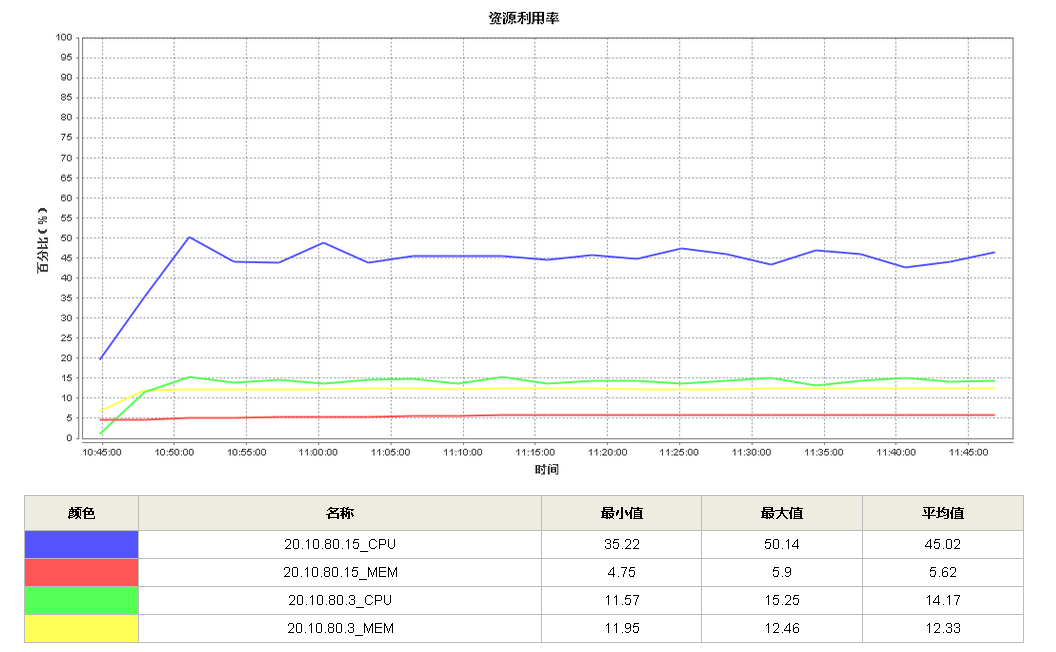
# 数据

**单人环境测试数据：**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 执行时间(s) | | | 数据库执行时间(s) | | |
|  | 复杂度 | 最小 | 最大 | 平均 | 最小 | 最大 | 平均 |
| 场景一：查询 | 轻 | 0.97 | 2.51 | 1.65 | 0.705 | 7.017 | 1.775 |
| 场景二：钻取 | 中 | 1.92 | 2.65 | 2.27 | 1.314 | 7.5 | 1.82 |
| 场景三：排序 | 中 | 9.84 | 10.96 | 10.44 | 13.657 | 18.593 | 15.013 |
| 场景四：时序分析（环比） | 中 | 3.39 | 5.20 | 4.54 | 2.361 | 10.5 | 5.781 |
| 场景四：时序分析（同比） | 中 | 3.31 | 5.63 | 4.34 | 2.406 | 12.608 | 5.608 |
| 场景五：切换指标 | 中 | 0.85 | 1.46 | 1.11 | 0.732 | 4.873 | 1.123 |
| 场景六：切换维度 | 中 | 2.56 | 4.46 | 3.87 | 2.705 | 7.483 | 4.131 |
| 场景七：结构化占比（父级占比） | 中 | 0.94 | 1.62 | 1.23 | 0.782 | 8.952 | 1.137 |
| 场景八：多个透视区域查询 | 重 | 15.64 | 17.62 | 16.31 | 24.313 | 63.063 | 46.767 |

**并发方案测试数据：**

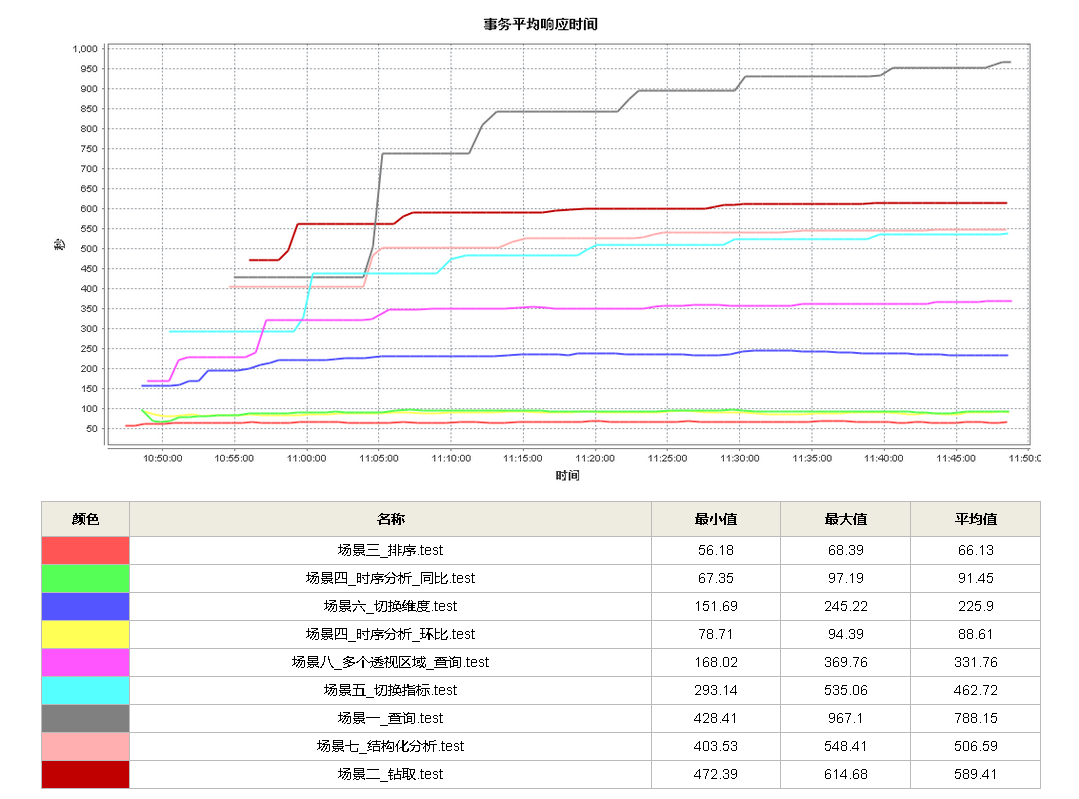
**1）资源利用率**

****

**2）每秒通过事务数**

****

**3）事务平均响应时间**

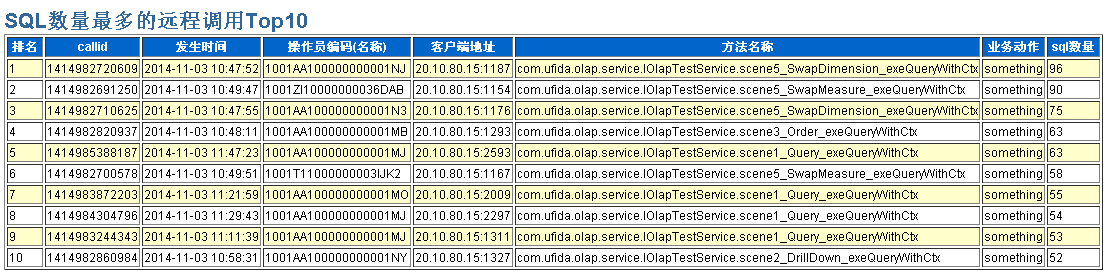
****

**4）事务执行次数**

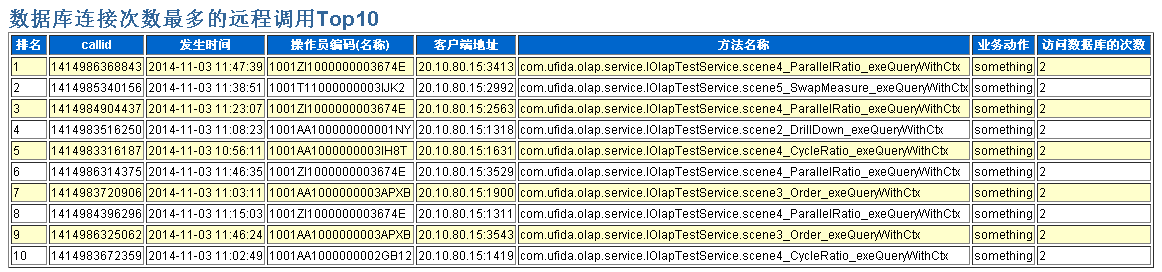
****

其中，成功事务总数：**362**，失败事务总数：**0**

**5）SQL数量最多的远程调用Top10**

****

**6）数据库连接次数最多的远程调用Top10**

****

# 附录

(基准测试配置文件的git地址，数据文件的备份目录)

1. **备份文件：**
2. 应用服务器和数据库服务器的虚拟机文件。
3. design数据源的dump文件，dw数据库的数据通过tpch生成。
4. 导出的数据仓库元数据、语义模型、分析主题、透视表的定义（通过design的dump文件可获得）。
5. 数据文件备份目录：\\20.2.80.19\olap性能基准测试文件备份
6. **Tpch数据生成步骤：**
7. 数据生成过程
8. 在官方网站(http://www.tpc.org/tpch/spec/tpch\_2\_17\_0.zip)下载数据生成工具包tpch\_2\_17\_0.zip
9. 配置/tpch\_2\_17\_0/dbgen/下面的makefile文件，需要配置的位置如下CC= gcc、DATABASE= ORACLE、MACHINE = LINUX、WORKLOAD = TPCH，参数依照具体的运行环境和需求而定
10. 在/tpch\_2\_17\_0/dbgen/目录下使用make指令对数据生成程序进行编译（如果makefile有后缀.suite需要重命名成无后缀的文件名），编译后会多出一些连接文件
11. 使用./dbgen –s 1进行数据的生成，其中1代表生成1G的数据量，其中数据量的取值只能取1, 10, 100, 300, 1000, 3000, 10000, 30000, 100000其中的一个，执行成功后，在/tpch\_2\_17\_0/dbgen/目录下会有8个表的对应的8 个以tbl为后缀的文件
12. 建立库表结构(使用如下sql创建相关的8张表结构)

CREATE TABLE NATION ( N\_NATIONKEY INTEGER NOT NULL,

N\_NAME CHAR(25) NOT NULL,

N\_REGIONKEY INTEGER NOT NULL,

N\_COMMENT VARCHAR(152));

CREATE TABLE REGION ( R\_REGIONKEY INTEGER NOT NULL,

R\_NAME CHAR(25) NOT NULL,

R\_COMMENT VARCHAR(152));

CREATE TABLE PART ( P\_PARTKEY INTEGER NOT NULL,

P\_NAME VARCHAR(55) NOT NULL,

P\_MFGR CHAR(25) NOT NULL,

P\_BRAND CHAR(10) NOT NULL,

P\_TYPE VARCHAR(25) NOT NULL,

P\_SIZE INTEGER NOT NULL,

P\_CONTAINER CHAR(10) NOT NULL,

P\_RETAILPRICE DECIMAL(15,2) NOT NULL,

P\_COMMENT VARCHAR(23) NOT NULL );

CREATE TABLE SUPPLIER ( S\_SUPPKEY INTEGER NOT NULL,

S\_NAME CHAR(25) NOT NULL,

S\_ADDRESS VARCHAR(40) NOT NULL,

S\_NATIONKEY INTEGER NOT NULL,

S\_PHONE CHAR(15) NOT NULL,

S\_ACCTBAL DECIMAL(15,2) NOT NULL,

S\_COMMENT VARCHAR(101) NOT NULL);

CREATE TABLE PARTSUPP ( PS\_PARTKEY INTEGER NOT NULL,

PS\_SUPPKEY INTEGER NOT NULL,

PS\_AVAILQTY INTEGER NOT NULL,

PS\_SUPPLYCOST DECIMAL(15,2) NOT NULL,

PS\_COMMENT VARCHAR(199) NOT NULL );

CREATE TABLE CUSTOMER ( C\_CUSTKEY INTEGER NOT NULL,

C\_NAME VARCHAR(25) NOT NULL,

C\_ADDRESS VARCHAR(40) NOT NULL,

C\_NATIONKEY INTEGER NOT NULL,

C\_PHONE CHAR(15) NOT NULL,

C\_ACCTBAL DECIMAL(15,2) NOT NULL,

C\_MKTSEGMENT CHAR(10) NOT NULL,

C\_COMMENT VARCHAR(117) NOT NULL);

CREATE TABLE ORDERS ( O\_ORDERKEY INTEGER NOT NULL,

O\_CUSTKEY INTEGER NOT NULL,

O\_ORDERSTATUS CHAR(1) NOT NULL,

O\_TOTALPRICE DECIMAL(15,2) NOT NULL,

O\_ORDERDATE CHAR(10) NOT NULL,

O\_ORDERPRIORITY CHAR(15) NOT NULL,

O\_CLERK CHAR(15) NOT NULL,

O\_SHIPPRIORITY INTEGER NOT NULL,

O\_COMMENT VARCHAR(79) NOT NULL);

CREATE TABLE LINEITEM ( L\_ORDERKEY INTEGER NOT NULL,

L\_PARTKEY INTEGER NOT NULL,

L\_SUPPKEY INTEGER NOT NULL,

L\_LINENUMBER INTEGER NOT NULL,

L\_QUANTITY DECIMAL(15,2) NOT NULL,

L\_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,

L\_DISCOUNT DECIMAL(15,2) NOT NULL,

L\_TAX DECIMAL(15,2) NOT NULL,

L\_RETURNFLAG CHAR(1) NOT NULL,

L\_LINESTATUS CHAR(1) NOT NULL,

L\_SHIPDATE CHAR(10) NOT NULL,

L\_COMMITDATE CHAR(10) NOT NULL,

L\_RECEIPTDATE CHAR(10) NOT NULL,

L\_SHIPINSTRUCT CHAR(25) NOT NULL,

L\_SHIPMODE CHAR(10) NOT NULL,

L\_COMMENT VARCHAR(44) NOT NULL);

1. 导入数据
2. 使用sqlldr name/password@ip:port/service control=/catalog1/load\_control.ctl进行表的数据导入过程，其中需要控制文件对数据的导入进行控制，控制文件的样例如下

load data

infile '/catalog2/customer.tbl'

append into table CUSTOMER

fields terminated by '|'

( C\_CUSTKEY ,

C\_NAME ,

C\_ADDRESS ,

C\_NATIONKEY ,

C\_PHONE ,

C\_ACCTBAL ,

C\_MKTSEGMENT ,

C\_COMMENT)

1. 建立主键、外键、索引
2. 建立主键

ALTER TABLE PART ADD PRIMARY KEY (P\_PARTKEY);

ALTER TABLE SUPPLIER ADD PRIMARY KEY (S\_SUPPKEY);

ALTER TABLE PARTSUPP ADD PRIMARY KEY (PS\_PARTKEY, PS\_SUPPKEY);

ALTER TABLE CUSTOMER ADD PRIMARY KEY (C\_CUSTKEY);

ALTER TABLE ORDERS ADD PRIMARY KEY (O\_ORDERKEY);

ALTER TABLE LINEITEM ADD PRIMARY KEY (L\_ORDERKEY, L\_LINENUMBER);

ALTER TABLE NATION ADD PRIMARY KEY (N\_NATIONKEY);

ALTER TABLE REGION ADD PRIMARY KEY (R\_REGIONKEY);

1. 建立外键

ALTER TABLE SUPPLIER ADD CONSTRAINT FK\_SUPPLIER\_1 FOREIGN KEY (S\_NATIONKEY) REFERENCES NATION(N\_NATIONKEY);

ALTER TABLE PARTSUPP ADD CONSTRAINT FK\_PARTSUPP\_1 FOREIGN KEY (PS\_PARTKEY) REFERENCES PART(P\_PARTKEY);

ALTER TABLE PARTSUPP ADD CONSTRAINT FK\_PARTSUPP\_2 FOREIGN KEY (PS\_SUPPKEY) REFERENCES SUPPLIER(S\_SUPPKEY);

ALTER TABLE CUSTOMER ADD CONSTRAINT FK\_CUSTOMER\_1 FOREIGN KEY (C\_NATIONKEY) REFERENCES NATION(N\_NATIONKEY);

ALTER TABLE ORDERS ADD CONSTRAINT FK\_ORDERS\_1 FOREIGN KEY (O\_CUSTKEY) REFERENCES CUSTOMER(C\_CUSTKEY);

ALTER TABLE LINEITEM ADD CONSTRAINT FK\_LINEITEM\_1 FOREIGN KEY (L\_ORDERKEY) REFERENCES ORDERS(O\_ORDERKEY);

ALTER TABLE LINEITEM ADD CONSTRAINT FK\_LINEITEM\_2 FOREIGN KEY (L\_PARTKEY,L\_SUPPKEY) REFERENCES PARTSUPP(PS\_PARTKEY,PS\_SUPPKEY);

ALTER TABLE NATION ADD CONSTRAINT FK\_NATION\_1 FOREIGN KEY (N\_REGIONKEY) REFERENCES REGION(R\_REGIONKEY);

1. 建立索引

CREATE INDEX IDX\_SUPPLIER\_NATION\_KEY ON SUPPLIER (S\_NATIONKEY);

CREATE INDEX IDX\_PARTSUPP\_PARTKEY ON PARTSUPP (PS\_PARTKEY);

CREATE INDEX IDX\_PARTSUPP\_SUPPKEY ON PARTSUPP (PS\_SUPPKEY);

CREATE INDEX IDX\_CUSTOMER\_NATIONKEY ON CUSTOMER (C\_NATIONKEY);

CREATE INDEX IDX\_ORDERS\_CUSTKEY ON ORDERS (O\_CUSTKEY);

CREATE INDEX IDX\_LINEITEM\_ORDERKEY ON LINEITEM (L\_ORDERKEY);

CREATE INDEX IDX\_LINEITEM\_PART\_SUPP ON LINEITEM (L\_PARTKEY,L\_SUPPKEY);

CREATE INDEX IDX\_NATION\_REGIONKEY ON NATION (N\_REGIONKEY);

-- aditional indexes

CREATE INDEX IDX\_LINEITEM\_SHIPDATE ON LINEITEM (L\_SHIPDATE, L\_DISCOUNT, L\_QUANTITY);

CREATE INDEX IDX\_ORDERS\_ORDERDATE ON ORDERS (O\_ORDERDATE);