速瑞 ArrayDB Beta 测试计划书

(Rev. 0.8, 9/1/2015)

Contents

简介 Introduction

本测试计划是为客户能进一步了解并确认速瑞公司(EXERAY)的 ArrayDB 新型数据库的功能和性能而制定。测试目的是发现和认识 ArrayDB 采用的与传统数据库不同的数据索引技术的性能，衡量新的数据库在数据录入、数据查询、并发处理方面的速度。

测试环境 Test Environment

系统将由 ArrayDB 服务器和客户端部分程序组成，服务器和客户端可以在不同主机下安装，也可以同一个主机内安装。

服务器端系统的运行要求

硬件要求：Pentium III 450 以上, 1024MB RAM, 200G HD
系统软件： Linux 2.6.23 CentOS, RedHat, Fedora, x86, x86_64
系统文件系统： ext4, 或者 XFS
测试过的系统： Fedora 3.1.0 x86_64, 8GB RAM, 512GB HD, ext4

客户端系统的运行要求

硬件要求：Pentium III 450 以上, 512MB RAM, 100G HD
系统软件： Linux 2.6.23 CentOS, RedHat, Fedora, x86, x86_64
系统文件系统：ext4, 或者 XFS

测试过的系统： Fedora 3.1.0 x86_64, 8GB RAM, 512GB HD, ext4

ArrayDB 系统安装

按照速瑞公司提供的软件包，可以实现一键安装。Serve、client、embedded 软件包可以分别安装在不同的主机或都安装到同一个主机上。ArrayDB 的服务器进程将在 TCP/IP 端口 8888 监听客户端发起的请求。ArrayDB 的服务器进程的名称是 rdbserv，此程序应该由 rdb 用户启动。服务进程 rdbserv 的启动程序脚本是 rdbstart.sh。停止运行的脚本程序是 rdbstop.sh。

ArrayDB Server 安装

在规定的服务器平台上，以 root 用户执行下列安装脚本程序：

```
# tar zxf   arraydb-server-n.n.tar.gz
```

有关文件和程序将解压到 arraydb-server-n.n 目录内。

```
# cd arraydb-server-n.n
# ./install.sh
```

安装程序将自动产生 rdb 操作系统用户，将配置文件 rdb.conf 拷贝到 /etc/rdb.conf，将 rdbserv and rdbadmin 拷贝到 /usr/bin 目录，还有两个启动和停止 ArrayDB server 服务器的脚本程序 rdbstart.sh， rdbstop.sh 也拷贝到 /usr/bin。

生成数据库和用户账户

安装好服务器程序后，可以转换到 rdb 用户账户，用 rdbadmin 程序产生数据库以及数据库用户账户。

切换到 rdb 系统账户后，执行下列程序产生数据库：

```
# su - rdb
$ rdbadmin createdb  mydb
```

其中是用户想要建立的数据库名称。利用 rdbadmin createdb 指令可以产生任意数据库。

使用 rdbadmin createuser 指令可以产生数据库用户账号。例如：

```
$ rdbadmin  createuser  tom:tom123
```

将建造用户名为 tom 密码为 tom123 的用户账号。一个新用户的默认权限只是具有读的功能，而不能对数据库进行写入和修改。若要给用户赋予数据读写的功能，必须执行下列命令：

```
$ rdbadmin  perm  tom:write
```

上述命令赋予了用户 tom 数据读和写的权限。

在产生用户的时候，用户的身份是一般用户。一般用户不能查看、修改数据库系统的数据。一般用户只有升级到管理员账户才有权限查询修改数据库系统的表格：

```
$ rdbadmin role  tom:admin
```

上述指令将 tom 升级到了管理员账户。

只是执行 rdbadmin 程序，其用法和解释将被列出：

```
$ rdbadmin
rdbadmin  command [argument]
rdbadmin  createdb    <databasename>         -- create a new database
rdbadmin  dropdb      <databasename>         -- drop a database
rdbadmin  createuser  <username:password>    -- create a new database user with
username and password
rdbadmin  dropuser    <username>             -- drop a database user
rdbadmin  role        <username:admin/user>  -- update role of user (admin or user)
```

rdbadmin perm <username:read/write> -- update permission of user (read: read only, write: read and write)

启动 ArrayDB Server

数据库和用户账户建立完毕后，切换到 rdb 系统账户启动 ArrayDB server:

```
# su - rdb
$ /usr/bin/rdbstart.sh
```

上述指令完成后，ArrayDB server 即在 port 8888 监听客户端发出的请求，执行数据库操作。在服务器启动后，rdb 用户仍可创建更多的数据库和用户账户。创建的数据库立即生效，但是创建的用户账户需要等待 10 分钟才能生效，被 rdbserv 采用。服务器的日志文件存入到 $RDB_HOME/system/log/ 目录内。其中 RDB_HOME 的值在文件/etc/rdb.conf 内指定。

ArrayDB Client 安装

在规定的客户端平台上，以 root 用户执行下列安装脚本程序:

```
# tar zxf arraydb-client-n.n.tar.gz
```

有关文件和程序将解压到 arraydb-client-n.n 目录内。

```
# cd arraydb-client-n.n
# ./install.sh
```

安装程序将配置文件 rdb.conf 拷贝到 /etc/rdb.conf, 将编程用的 API rdbclient.h 拷贝到 /usr/include, 将编程用的 API 函数库 librdbclient.a 拷贝到 /usr/lib, 将交互式客户端 rdb 程序拷贝到 /usr/bin 目录内。

文件 /etc/rdb.conf 包含下列参数:

```
RDB_HOME = /home/rdb/database
```

port=8888

参数 RDB_HOME 是 ArrayDB 数据库系统存放所有数据的总目录，只是对服务器适用。Port 8888 指定了服务器监听的 port 数字。

ArrayDB Embedded Database 安装

ArrayDB 内嵌式（embedded or stand alone）数据库与上述 server 和 client 软件包不同。它直接产生底层数据文件，写入和查询都在此数据文件上操作，没有通过 client-server 网络层的指令和数据的传输。ArrayDB server 内核的数据和索引操作就是基于此内嵌式数据库模块。

安装时在规定的客户端平台上，以 root 用户执行下列安装脚本程序：

```
# tar zxf   arraydb-embed-n.n.tar.gz
```

有关文件和程序将解压到 arraydb-embed-n.n 目录内。

```
# cd arraydb-embed-n.n
# ./install.sh
```

安装程序将头文件 ArrayDB.h 以及相关头文件 rdb.conf 拷贝到 /usr/include，将编程用的 API 函数库 libarraydb.a 拷贝到 /usr/lib。

此软件包还提供了一个 C++实例程序，rdbtest.cc，显示如何使用 ArrayDB C++ class 产生 array 索引文件，写入数据，和查询数据。他同时也可以作为 benchmark ArrayDB server 内核性能的测试程序。

条件和限制 Platform Provisioning

Mount noatime

数据库系统运行时，IO 的性能是非常重要的因素。因此建议关闭文件系统的 access time 功能。具体操作步骤是 root 编辑 /etc/fstab 文件在默认选项值内 加入 noatime:

defaults,noatime

安装确认 Installation Verification

ArrayDB Server 安装完毕后，确认下列文件设置正常：

1) Port 8888 没有与其它服务或应用程序冲突
2) 用户 rdb 的主目录/home/rdb 成功建立
3) /etc/rdb.conf RDB_HOME 指向有效目录
4) 下列文件存在并且文件模式正确（可执行）：

/usr/bin/rdbserv
/usr/bin/rdbadmin
/usr/bin/rdbstart.sh
/usr/bin/rdbstop.sh

ArrayDB Client 安装完毕后，确认下列文件正常存在：

/usr/bin/rdb

/usr/include/rdbclient.h

/usr/lib/librdbclient.a

ArrayDB Embed 安装完毕后，确认下列文件正常存在：

/usr/include/ArrayDB.h

/usr/include/abax.h

/usr/include/AbaxFormat.h

/usr/lib/libarraydb.a

测试执行 Test Execution

测试途径 Test Approaches

针对 ArrayDB 的测试可以通过三种途径：1）交互式的 Client-Server 会话 2）通过利用编程 API 方式 3）内核数据文件的操作。第一重方式启用 rdb 客户端程序进行，测试用户输入 RQL（类似 SQL 的语言）语句，server 接收到指令后，执行相关操作。第二种方式， 由测试用户或开发人员利用我们提供的 C API，编写 C 语言程序，请求数据库服务器进行数据写入、查询操作。第三中方式利用 ArrayDB C class 通过编程测试 server 内核对底层数据文件的写入和查询等操作。

基本性能测试 Baseline Performance (测试阶段一)

第一阶段的测试针对 ArrayDB 的优势部分进行测试，不需要测试产品的完备性和稳定性。测试方面包括一些几个基本性能：

- 数据插入速度
- 数据查询速度
- 索引性能
- 内存用量
- Join 的性能
- 在多用户并发情况下上述的性能

数据插入速度

针对数据插入速度的测试有三种方式：server 进行 batch load；client 连接 server 进行单个的数据插入；在内嵌数据文件中直接写入数据。在测试之前应该保证 server, client, embed 软件包都已经正确安装，并且正常运行。

预备工作：

1. 建立用户

```
# su - rdb
$ rdbadmin createuser test:test
$ rdbadmin perm test:write
```

2. 建立表格

```
$ rdb -u test -p test -d test -h 127.0.0.1:8888

rdb> create table test ( key: uid char(16), value: addr char(16) );
```

(A) Batch Load

在 ArrayDB server 主机上可以测试 load 3 百万条数据。样品数据可以由 client 软件包附带的 genrand 程序产生:

```
$ genrand 3000000 1 uid addr
$ mv genrand.out /tmp/3M.txt
```

然后在 rdb 客户端（任何用户可以运行 rdb 客户端）输入下列指令将 3 百万条数据从 /tmp/3M.txt 插入到 test 表格中:

```
rdb> load file /tmp/3M.txt into test format S;
```

预期值: 等待一段时间后（大约 2~3 分钟），rdb 将会报告 load 数据占用了多少 milliseconds (毫秒)。

(B) Client 单个数据插入

Client 的软件包附带的 rbench 程序用来实现从客户端向 server 插入、修改、查询数据。下列指令将产生 10000 个随机数，连接 server，数据传输给 server，server 将数据插入到数据库的 test 表内:

```
$ rbench -t test -r “10000:0:0” -k 0 | tee -a test.log
```

其中双引号内以冒号分隔的数字第一个是插入次数，第二个是修改次数，第三个是查询次数。 选项“-k 0”代表 test 数据表清空且重新产生。

预期值: 利用 rbench 插入数据，速度大约为 6000-8000 个/秒。rbench 包括的网络数据传输的 overhead。

(C) 内嵌数据文件直接写入

使用 Embed 软件包中附带的 rdbtest 程序，对数据库文件直接写入数据：

```
$ ./rdbtest 3000000 | tee -a test.log
```

结果将会报告插入数据用时：

```
Successfully insert()  3000000 records in nnnn milliseconds  IOPS=38235
```

预期值：rdbtest 插入速度大约为 30000 - 60000 个/秒。

数据查询速度

(A) rbench 客户端测试查询速度

```
$ rbench -t test -r "0:0:10000" -k 0 | tee -a test.log
```

上述指令将对数据库 server 查询 10000 次。

预期值：利用 rbench 查询数据，速度大约为 4000-8000 个/秒。rbench 涉及的网络 overhead 比较 rdbtest 多一些。
待做工作：后端数据 parse,网络传输可以更进一步改进，提高速度。

(B) rdbtest 测试 server 内核数据查询速度

```
$ rdbtest 3000000 | tee -a test.log
```

```
exist():      查询所有插入的数据是否存在
getValue():   对所有插入的 Key 读出 Value 数据
scanned:      将插入的数据从头到尾全部读出 (对累计、分析尤其有用)
```

预期值：

exist(): 每秒查询次数在 50 万-70 万

getValue(): 每秒查询次数在 40 万-50 万

Index Scan: 每秒查询个数在 5000 万-8000 万

索引性能

在 rdbtest 内嵌数据库测试中，exist()和 getValue()是通过哈希表进行。Scan 是表索引的数据扫描。IOPS 的值是每秒内完成的操作。

通过哈希表索引的查询速度大约在 50 万-70 万次/秒。索引 scan 的速度大约为 5000 万-8000 万数据个数/秒。

内存用量

在进行数据 load 过程中，通过 top 指令可以监测 server 进程 rdbserv 的内存用量。

```
$ top
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+  COMMAND
12360 rdb       20   0    108m    26m    904 S  99.4   0.3   0:09.02 rdbserv
```

主要观测 RES 内存的数值，这个数值是程序真正耗用的内存数量。

使用 rdbtest 对内嵌数据库测试，结果会打印出内存用量：

```
$ rdbtest 1000000
Memory used:             14 MB
```

ArrayDB 使用内存非常少，省去的内存可以用来做数据分析， 以及复杂的计算等任务，整体上更进一步提高数据库服务器的性能。

Join 的性能

对两个或更多个 table 的 Join 可以通过下面简单步骤实现：

1) 使用 genrand 产生 几个 1000 万数据的文件

```
$ genrand 10000000 1 uid addr
$ mv genrand.out /tmp/10M1.txt

$ genrand 10000000 1 uid phone
$ mv genrand.out /tmp/10M2.txt

$ genrand 10000000 1 uid email
$ mv genrand.out /tmp/10M3.txt
```

2) 建立表格

```
rdb> create table t1 ( key: uid char(16), value: addr char(16) );
rdb> create table t2 ( key: uid char(16), value: phone char(16) );
rdb> create table t3 ( key: uid char(16), value: email char(16) );
```

3) Load 数据

```
rdb> load file /tmp/10M1.txt into t1 format S;
rdb> load file /tmp/10M2.txt into t2 format S;
rdb> load file /tmp/10M1.txt into t3 format S;
```

Load 数据会需要一段时间。每次 load 需要大概 2-3 分钟。

4) Join

```
rdb> select * join ( t1, t2, t3 ) limit 10;
```

此指令将 Join 三个 table, 选出 Key 部分都相同的 row,将数据打印出来。进行此操作时, 可以利用 top 指令观察 rdbserv 内存的使用情况。

预期值：0.9 秒之内三个 300 万的表 Join 会完成, 结果会打印出来。

在多用户并发情况下上述的性能

使用 rbench 程序能模拟多用户并发连接和请求。在多个客户端发起请求下，测试 ArrayDB 服务器的响应速度和 ArrayDB 数据库的性能。测试工作包括以下内容:

1) 多个客户端的 100%数据写入操作。
2) 多个客户端的 100%数据查询操作。
3) 多个客户端的 20%数据写入操作，80%数据查询操作。

100%数据写入操作:

```
$ rbench -t test -r "10000:0:0" -k 0 -c 4 | tee -a test.log
```

此处"-c 4"指定同时启动 4 个线程，每个线程向 server 发出 10000 次数据写入操作。实际启动的并发线程数目应当与服务器的 core 数目一致。例如服务器有 16 个 core，并发线程数目应该为 16，以保证达到最大数据写入速度。

100%数据查询操作:

```
$ rbench -t test -r "0:0:10000" -c 4 | tee -a test.log
```

此处"-c 4"指定同时启动 4 个线程，每个线程向 server 发出 10000 次数据查询操作。实际启动的并发线程数目应当与服务器的 core 数目一致。例如服务器有 16 个 core，并发线程数目应该为 16，以保证达到最大数据查询速度。

20%数据写入操作，80%数据查询操作:

```
$ rbench -t test -r "2000:0:8000" -c 4 | tee -a test.log
```

此处"-c 4"指定同时启动 4 个线程，每个线程向 server 发出 2000 此数据写入操作，8000 次数据查询操作。如上所述，实际启动的并发线程数目应当与服务器的 core 数目一致。

在多用户并发请求情况下，可以利用 top 指令观察 rdbserv 的内存占用情况。

预期值：每增加一个并发线程，性能会提高。但是当线程的数目多于主机的实际 core 数目时，速度将不再会增加。

备注：rdbserv 服务程序现在为止对多用户并发处理实行简单的全部文件锁。后端开发人员计划进行文件局部锁（write and read lock）。等完成开发局部锁后， 并发处理速度会更大提高。

基本功能测试 Baseline Functionality (测试阶段二)

下列基本功能可以通过上述两种测试途径完成：

1) create database 创建新的数据库
2) create user 创建新用户，改变权限，身份
3) create table 创建表格
4) insert data into table 写入数据
5) select data from table 查询数据
6) update data in table 更新数据
7) delete data from table 删除数据
8) join tables and select data 交互查询多个表格
9) load file into table 批量 load 数据
10) drop a table 删除表格
11) drop database 删除数据库

利用会话式客户端 rdb 上述测试的方法通过 help 指令可以得到帮助：

```
rdb> help
You can enter the following commands:
help use            (how to use databases)
```

```
help desc          (how to describe tables)
help show          (how to show tables)
help create        (how to create tables)
help insert        (how to insert data)
help load          (how to batch load data)
help select        (how to select data)
help update        (how to update data)
help delete        (how to delete data)
help join          (how to join two or more tables)
help drop          (how to drop a table completely)
```

Kingbase 要求的测试 (测试阶段二)

根据客户的特殊条件，可以进行 C API 编程的测试。客户和速瑞开发人员可以共同协作进行。

测试数据收集和汇总 Data Collection

测试情况可能很多，对测试结果的数据收集和归纳就很重要。我们计划对测试程序模块化、自动化， 数据搜集也做到完成，容易查询、观察。

1) 模块化：每个模块测试比较独立的功能或性能，并且采用输入参数
2) 自动化：每一个模块进行测试时，相关的初始条件已经具备
3) 结果查询：测试环境、输入参数、结果保存、分类到文件或数据库，便于查询
4) 预期性能：测试的结果应该具备预期结果，结果与预期值相差较大时，采取措施。

对每一个测试 Cycle，做到记录和标签（label）下列数据：

1) 原始数据、输入数据的记录和保存
2) 测试参数的标记和保存
3) 输入数据以及测试结果数据容易分类、归纳

对测试结果应该有合适方式（脚本语言、数据库、图表等工具）方便查询和分析。

附录：MySQL 的对比测试

下面内容提供测试 MySQL 数据库性能的参考方法。

MySQL Load 数据

向 MySQL 数据库添加数据之前，可以使用 genrand 产生数据文件：

```
$ genrand 10000000 1 uid addr
$ mv genrand.out  /tmp/10M_MYSQL.txt
```

然后建立表：

```
mysql> create table t1 ( uid varchar(16) primary key, addr varchar(16) );
```

数据可以按照如下指令添加到表中：

```
mysql> load data infile '/tmp/10M_MYSQL.txt' into table t1 fields
terminated by ',' lines terminated by '\n';
```

在此过程中用 top 观察 mysqld 的内存用量。

MySQL Insert 数据

MySQL client 软件包提供 C 编程 API，让 C 程序连接 MySQL server 并且插入查询数据。

下面的一段 C 程序显示如何对 MySQL 数据库插入 1000 条数据：

```
mysql_real_connect(con, "localhost", username, passwd, "test", 0, NULL, 0);
for ( i = 0; i < 1000 ; ++i )
{
```

```
    k = randomString(16);
    v = randomString(16);
    sprintf( sqlcmd, "insert into %s values ('%s', '%s' ) ", table, k, v );
    mysql_query( g_mysqlcon, sqlcmd );
}
```

MySQL Select 数据

下面的一段 C 程序显示如何对 MySQL 数据库查询 10000 条数据：

```
    for ( i = 0; i < 10000; ++i )
    {
        k = randomString( 16 );
        v = randomString( 16 );
        sprintf( sqlcmd, "select * from %s where uid='%s'", table, k );
        mysql_query( con, sqlcmd );
        result = mysql_store_result( con );
    }
```

MySQL Join Tables

通过程序产生原始数据，然后再将这三个文件导入到三个表中：

```
   mysql> create table t1 ( uid varchar(16) primary key, addr varchar(16) );
   mysql> create table t2 ( uid varchar(16) primary key, phone varchar(16) );
   mysql> create table t3 ( uid varchar(16) primary key, email varchar(16) );
   mysql> load data infile '/tmp/10M_MYSQL_1.txt' into table t1 fields
terminated by ',' lines terminated by '\n';
   mysql> load data infile '/tmp/10M_MYSQL_2.txt' into table t2 fields
terminated by ',' lines terminated by '\n';
```

```
   mysql> load data infile '/tmp/10M_MYSQL_3.txt' into table t3 fields
terminated by ',' lines terminated by '\n';
```

Load MySQL 1000 万数据大约需要 1-3 小时不等。

最后执行 JOIN 操作:

```
   mysql> select * from t1, t2, t3 where t1.uid = t2.uid and t2.uid = t3.uid;
```

MySQL 会在完成操作后报告 Join 过程花费的时间。