LSST:UK Consortium LSST:UK Science Centre

Initial Assessment of Qserv for LSST:UK

Work Package 2.4

Project Acronym	LUSC-A
Project Title	UK Involvement in the Large Synoptic Survey Telescope
Document Number	LUSC-A-07

Submission date	6/NOV/19
Version	1.0
Status	Final
Author(s) inc. institutional affiliation	Mike Read (Edinburgh), Darren White (Edinburgh), Bob Mann (Edinburgh) Teng Li (Edinburgh)
Reviewer(s)	George Beckett (Edinburgh), Bob Mann (Edinburgh)

Dissemination level	
Public	*Once finalised, it can be distributed without restriction*

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	20/FEB/19	Initial version	MAR
0.2	21/FEB/19	Added DW Queries	DJW
0.3	7/FEB/19	Added brief description of query sources	DJW
0.3.1	8/FEB/19	Minor update, some formatting	DJW
0.3.2	11/MAR/19	Minor update to Introduction	RGM
0.4	17/SEP/19	Added information on SQL Server-to-MySQL conversion tool	ETWS
0.5	31/OCT/19	Fixing formatting and proofing text	MGB
0.51	5/NOV/19	Minor updates	RGM
1.0	6/NOV/19	Version 1.0 published	MGB

Table of Contents

Index of Figures

1 Executive Summary

This report summarises an initial assessment of Qserv for LSST:UK conducted by the UK DAC team as part of the LSST:UK Science Centre (LUSC) Phase A programme.

We have successfully set up three different Qserv configurations (single-node, multi-node, containerised) and ingested sky survey catalogue data from UKIDSS and SDSS, stored in the WFCAM Science Archive (WSA[1]) and with which the DAC team are familiar.

This process was complicated by several factors, some of which may apply to the ingestion of any non-LSST catalogues.

- Firstly, the WSA is implemented in Microsoft SQL Server, necessitating some schema and data conversion to generate a MySQL-compliant version that could be loaded into Qserv.
- Secondly, the WSA makes use of (out-of-range) default values, rather than nulls, and foreign key constraints require the existence in some tables of default rows, composed entirely of default values; Qserv does not follow that design choice, so some rows had to be removed from WSA tables prior to ingest, and, hence, the version of UKIDSS DR8 implemented in Qserv is not identical to the original in the WSA.
- Thirdly, the spatial partitioning scheme adopted by Qserv assumes specific (director-child) relationships between some tables that may not have exact analogues in existing sky survey catalogues.

For a multi-node Qserv instance, the source and object tables needed to be "chunked" – this is intended when using Qserv. Several issues were encountered and worked around to support chunked versions of the catalogues. The lessons learned are encapsulated in a set of scripts that go a significant way towards automating the ingestion process.

Having ingested UKIDSS and SDSS catalogues into Qserv, we then tested Qserv's effectiveness and efficiency with a range of illustrative and realistic queries. Known limitations in Qserv (e.g., lack of neighbour tables and views) inhibited or prevented the successful implementation of some important queries. Further, the Qserv software was found to be fragile, subject to a number of (sometimes intermittent) failures and crashes.

Despite problems, we were able to successfully implement a little over half of the example queries (16 out of 27). Performance was variable (running on comparable hardware): sometimes there was little difference between Qserv and the original Microsoft SQL Server implementations; sometimes Qserv was significantly slower; and occasionally Qserv was faster.

Further study is required to understand performance discrepancies, and the results reported here are considered preliminary, with a more detailed performance evaluation being deferred until Qserv is more mature, exhibiting better stability and incorporating crucial missing functionalities.

In follow-on work it would be appropriate to further refine the ingestion process and aim to expose a working Qserv instance to a small number of early-adopter astronomers, to gain experience of the support of Qserv (e.g., software-update process, performance under multi-user load) and to begin to disseminate Qserv interface characteristics to those communities who will likely rely on Qserv for their LSST-related research programmes.

[1] wsa.roe.ac.uk

2 Introduction

2.1 Background

The UK DAC team, based in the Wide-Field Astronomy Unit (WFAU) at the University of Edinburgh, operates a number of sky survey archives. These include the WFCAM Science Archive (WSA, wsa.roe.ac.uk) for the UK Infrared Deep Sky Survey (UKIDSS, www.ukidss.org) and the VISTA Science Archive (VSA, vsa.roe.ac.uk), which hosts data from most of the near-infrared imaging surveys undertaken with VISTA/VIRCAM as part of the ESO Public Surveys programme. These science archives are all currently implemented in Microsoft SQL Server.

Recognising an expectation that single-server systems will not cope with catalogues of the scale of those from LSST, the LSST Data Management team has been developing a distributed-database system called Qserv [https://ldm-135.lsst.io/v/DM-5035/]. LSST:UK plans to operate a Data Access Centre (DAC), which will include catalogues from LSST, plus complementary surveys to support multi-wavelength analyses. In order to gain familiarity with Qserv and to assess its route to meeting UK science requirements, the DAC team undertook an initial study of Qserv during LSST:UK Science Centre (LUSC) Phase A. This report presents the results of that initial assessment, which covers:

(i) installation of several instances of Qserv with different configurations
(ii) ingestion of UKIDSS and SDSS data into Qserv
(iii) execution of a set of benchmark queries, designed to exercise the functionality of Qserv and assess its performance for common query types relative to that of existing WFAU sky survey archives.

3 Qserv

3.1 Introduction

LSST will undertake, from late 2022, a ten-year survey of the dynamic universe. During operation, the telescope will map the entire southern sky every few nights, taking ~2,000 exposures per operating night, which will generate a total data volume of ~16 TB daily. After processing, the detected objects will be stored in database catalogues estimated to include ~37 billion stars and galaxies, constituting a dataset at the scale of tens of PB. To manage such a huge volume of catalogues and provide query services for the worldwide astronomy community, a brand-new DBMS (Database Management System) is needed which must fulfil the following requirements: incremental scaling, near real-time response time for ad-hoc simple user queries, fast turnaround for full-sky scans/correlations, reliability, and low cost.

To satisfy the needs mentioned above, the LSST Data Management team is developing a distributed database system, Qserv, whose architecture and basic functionalities are shown as below.

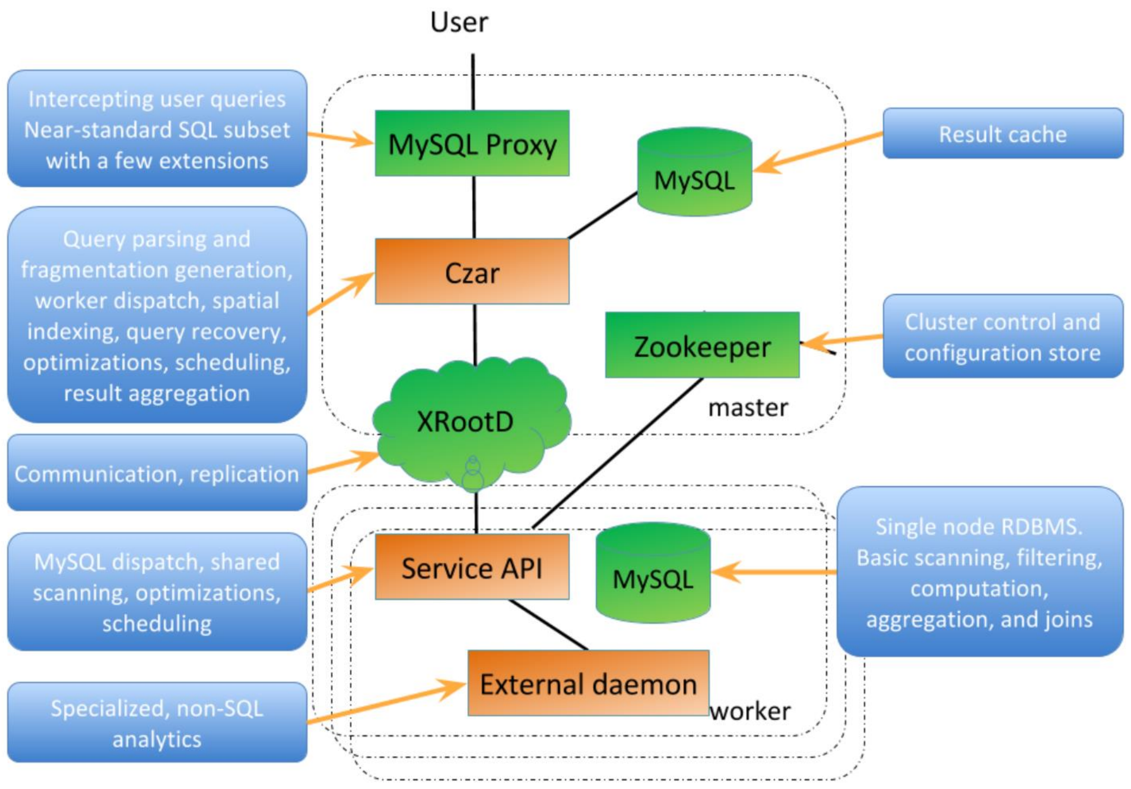
User
Intercepting user queries Near-standard SQL subset with a few extensions
MySQL Proxy
MySQL
Result cache
Query parsing and fragmentation generation, worker dispatch, spatial indexing, query recovery, optimizations, scheduling, result aggregation
Czar
Zookeeper
Cluster control and configuration store
XRootD
master
Communication, replication
Service API
MySQL
Single node RDBMS. Basic scanning, filtering, computation, aggregation, and joins
MySQL dispatch, shared scanning, optimizations, scheduling
External daemon
worker
Specialized, non-SQL analytics

Figure 1: Qserv High-level Architecture [2].

Qserv is essentially a multi-processor and parallel relational database running on a share-nothing server cluster. The large catalogues can be partitioned and distributed across worker nodes according to the sky coordinates scheme of the objects, while smaller catalogues can be replicated on each server. The user queries are firstly processed on the master node, where they are evaluated and re-written by the Czar process (see figure) and then submitted to the worker nodes to be executed in parallel, with the results being served back and merged on the master node.

[2] J. Becla et al., LDM-135: Database design, https://ldm-135.lsst.io/v/DM-5035/

3.2 Test-bed deployment

To test Qserv, a progressively more complex sequence of installations has been proposed, which begins with what we believe is the simplest possible installation and ends with a (very) small-scale version of what is envisaged to be required for a Data Access Centre.

First, a Qserv installation based on a single node was made alongside, on an equivalent hardware configuration, a stand-alone MySQL installation. This was done to allow a performance comparison of Qserv, which uses a MySQL installation as the backend, against a plain MySQL installation, and to have a performance data point for a single node. This information can then be used for tuning of the parameters and to measure the speedup achieved on multi-node installations.

The second installation is based on a two-node configuration, where one node is configured as the master node and the other as the worker node. The aim of that is to compare the performance of a mono-node Qserv and a multi-node Qserv to see if there is performance speedup or overhead. This is also to test if the queries work with the partitioned tables, and if not, to study the way to rewrite the queries efficiently to fulfil the request of Qserv.

The third installation is based on a (Docker) containerised deployment, with three containers deployed (on one physical machine), one served as the master node and two as worker nodes. The Docker-based installation is made to test functionalities of containerized Qserv, which could be a common case for running large-scale production. The second and third installations were also made as test-beds for developing partitioning and ingesting tools, to be discussed in Section 3.3.

3.2.1 Test-bed configuration

For the testbed, two machines have been set up with the following configuration:

- 2 Intel Xeon E5-2600v4 CPUs, 2x 8cores 20MB cache (16 cores with HT)
- Intel X540 Dual 10Gbps LAN card
- 128 GB RAM (16x8GB, DDR4 2133MHz)
- 2x Intel DC3510 SSD, 120GB
- 12x Hitachi 7200rpm SAS HDD, 4TB, 12Gb/s

All machines are connected to the same network switch with a 1Gbps connection. They are reachable on that port from other machines and from the public Internet for data transfers or queries. In addition, a second network port is used to connect the 2 machines with each other through a 10Gbps link. This link is only usable within the test-bed environment.

For the OS, CentOS Linux release 7.3 is installed in a minimal configuration. The disks are initially configured as software RAID based on ZFS. For all installations, a raidz2 configuration, similar to raid6, is used with 9 data disks and 1 spare disk, with compression enabled.

```
[root@lsstuk1 ~]# zpool status
pool: datapool
state: ONLINE
scan: scrub repaired 0 in 2h36m with 0 errors on Tue Jan
24 14:47:25 2017
config:
NAME        STATE     READ WRITE CKSUM
datapool    ONLINE       0     0     0
raidz2-0  ONLINE       0     0     0
sdc     ONLINE       0     0     0
```

```
sdd     ONLINE       0     0     0
sde     ONLINE       0     0     0
sdf     ONLINE       0     0     0
sdg     ONLINE       0     0     0
sdh     ONLINE       0     0     0
sdi     ONLINE       0     0     0
sdj     ONLINE       0     0     0
sdk     ONLINE       0     0     0
sdl     ONLINE       0     0     0
sdm     ONLINE       0     0     0
spares
sdn       AVAIL
errors: No known data errors
```

In order to install Qserv, a set of basic libraries, as well as the LSST software stack, need to be installed. This is documented in the Qserv installation instructions. The procedures are listed as a command-line history:

1. As root, install system dependencies and enable devtoolset-7

```
yum install \
bison \
blas \
bzip2 \
bzip2-devel \
cmake \
curl \
flex \
fontconfig \
freetype-devel \
gawk \
gcc-c++ \
gcc-gfortran \
gettext \
git \
glib2-devel \
java-1.8.0-openjdk \
libcurl-devel \
libuuid-devel \
libXext \
libXrender \
libXt-devel \
make \
mesa-libGL \
ncurses-devel \
openssl-devel \
patch \
perl \
perl-ExtUtils-MakeMaker \
readline-devel \
sed \
tar \
which \
zlib-devel
yum install centos-release-scl
yum install devtoolset-7-gcc-gfortran devtoolset-7-gcc
devtoolset-7-gcc-c++
scl enable devtoolset-7 bash
```

2. Install LSST stack as a non-root user (qserv)

```
NEWINSTALL_URL=https://raw.githubusercontent.com/lsst/
lsst/master/scripts/newinstall.sh
mkdir /home/qserv/lsst_stack
INSTALL_DIR=/home/qserv/lsst_stack
cd $INSTALL_DIR
curl -OL ${NEWINSTALL_URL}
bash newinstall.sh
. loadLSST.bash
```

3. Install qserv_distrib as qserv user

```
RELEASE="qserv-dev"
eups distrib install --tag $RELEASE qserv_distrib
setup qserv_distrib --tag $RELEASE
```

The LSST software stack uses EUPS to manage software releases. The release tags are listed in the official repository, https://eups.lsst.codes/stack/src/tags/. The latest released Qserv was used during testing, which is defined in the tag "qserv-dev". Since Qserv was evolving quickly during the tests, it was updated several times for bug fixing. The final version of Qserv distribution, which is the version evaluated in this report is labelled "2016_08-1-g76ecee9+202".

3.2.2 Single-node configuration

The Single-node configuration means that the data partitioning functionality is disabled. After the LSST software stack was installed, the following command finished the configuration of a single-node Qserv.

```
qserv-configure.py -R /datapool/qserv
```

To start Qserv or to have access to its control scripts, the environment script:

```
/home/qserv/stack/loadLSST.bash
```

—needs to be executed. Run `/datapool/qserv_head/bin/qserv-start.sh` to start the Qserv service. To access the database behind Qserv, port 13306 is used.

For the standalone MySQL installation, MariaDB from the official repository was used. MySQL is configured to be accessed on its default port.

The database configuration file for Qserv and MySQL is `/datapool/qserv/etc/my.cnf`, and `/etc/my.cnf`, respectively. These configurations are not fully optimized and there are many more possibilities to tune each configuration to a specific workload/ host platform. The chosen configuration is shown below, and needs to be the same on both systems for benchmarking against each other.

```
[mysqld]
port=3360
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
```

```
#  Disabling  symbolic-links  is  recommended  to  prevent
assorted security risks
symbolic-links=1
bind-address=0.0.0.0

default-storage-engine = MyISAM

key_buffer_size=16G

#  Disabling  symbolic-links  is  recommended  to  prevent
assorted security risks
symbolic-links=0

# In order to avoid "table is full" error
tmp_table_size=4G
max_heap_table_size=4G

max-connections                = 512

# enable InnoDB support via plugin

ignore-builtin-innodb
plugin-load=innodb=ha_innodb.so
innodb_file_per_table=1
tmpdir=/datapool/tmp

#  Settings  user  and  group  are  ignored  when  systemd  is
used.
#  If  you  need  to  run  mysqld  under  a  different  user  or
group,
# customize your systemd unit file for mariadb according
to the
# instructions in http://fedoraproject.org/wiki/Systemd

[mysqld_safe]
log-error=/var/log/mariadb/mariadb.log
#general-log=/var/log/mariadb/general.log
pid-file=/var/run/mariadb/mariadb.pid

#
# include all files from the config directory
#
!includedir /etc/my.cnf.d
```

3.2.3 Two-node configuration

In the two-node configuration, the host `lsstuk1.roe.ac.uk` serves as the master node, while `lsstuk4.roe.ac.uk` serves as the worker node. The procedures for configuration are listed as followed:

1. Generate the work directory for Qserv on both nodes:

```
qserv-configure.py -i -R /datapool/qserv_head
# For worker node use /datapool/qserv_worker
vim /datapool/qserv_head/qserv-meta.conf
# Edit qserv-meta.conf, change the "node
# type" to "master" for head node or "worker" for worker
# nodes. Change "master" to "lsstuk1"
qserv-configure.py -R /datapool/qserv_head
# For worker node use /datapool/qserv_worker
```

2. Copy the secret file to the worker node

```
scp \
  lsstuk1.roe.ac.uk:/datapool/qserv_head/etc/wmgr.secret
  /datapool/qserv_worker/etc/wmgr.secret
```

3. Start the Qserv services on both nodes

```
/datapool/qserv_head/bin/qserv-start.sh
```

4. Configure the CSS (Central State System) on the master node

```
qserv-admin.py "CREATE NODE WORKER2 type=worker \
port=5012 host=lsstuk1"
```

The database configuration is the same as the single-node installation.

3.2.4 Containerised installation

Qserv can be deployed within Docker containers. In this test, a three-node containerised Qserv instance was deployed on lsstuk1.roe.ac.uk. Here is the procedure of the deployment:

```
git clone https://github.com/lsst/qserv.git
cd  qserv/admin/tools/docker/deployment/localhost
cp env.example.sh env.sh
. run-multinode-tests.sh
```

Qserv containers, including the master node and worker nodes are deployed on the machine, and unit tests (basic functionality tests, implemented in Qserv code, for operations such as: create database/ table and run simple query) is performed on them. In the env.sh, parameters like numbers of worker nodes, DNS name of the containers and version of Qserv can be defined. Here is the one used in this test:

```
VERSION=dev

NB_WORKERS=3

# Set nodes names
DNS_DOMAIN=localdomain
MASTER=master."$DNS_DOMAIN"

for i in $(seq 1 "$NB_WORKERS");
do
    WORKERS="$WORKERS worker${i}.$DNS_DOMAIN"
done

# Set images names
MASTER_IMAGE="qserv/qserv:${VERSION}_master"
WORKER_IMAGE="qserv/qserv:${VERSION}_worker"
```

3.3 Data ingestion

Qserv is a distributed share-nothing database designed to satisfy spatial self-joining and cross-matching queries at a scale of trillions of detections. To achieve that, large catalogues are partitioned using a spatial partition scheme and evenly distributed across multiple worker nodes.

There are three kinds of tables in Qserv, *director* tables, *child* tables and *non-partitioned* tables. The non-partitioned tables will be entirely replicated across all worker nodes, suitable for small tables or tables that cannot be partitioned. The director (or dominant) table contains the sky coordinates used for partitioning the data between worker nodes. For an LSST Data Release, the "Object" table is chosen as the director table and "ra_ps" and "decl_ps" columns are used as the coordinates for partitioning. Child tables are paired with the director table, via the Object Id, which is a foreign key. Put simply, each source in the source catalogue has an ObjectID attribute, which is guaranteed to correspond to an entry in the Object table (the Director). In Qserv, Director and Child tables have to be partitioned to ensure that all entries in each child table reside on the same worker node as their corresponding entry in the Director table.

To test the functionalities and performance of Qserv, we ingested a UKIDSS data release into Qserv. We partitioned the source and detection tables (gpsSource, gpsDetection, dxsSource, dxsDetection, lasSource, lasDetection, gcsSource and gcsDetection) as separate director tables, since they are large, and each contains sky coordinates columns which can be used for partitioning. The remaining UKIDSS tables are not partitioned and simply replicated to all worker nodes.

Since Qserv did not, at the time of writing, provide an automated data ingestion system, some of the ingestion procedures had to be done manually.

The data ingestion can be divided into three steps: data preparation, data ingesting and metadata registration.

3.3.1 Data preparation

UKIDSS data releases are implemented within a Microsoft SQL Server account database in the WSA. As a first step, we needed to port the data into a format that was readable by MySQL. The schema needed to be updated first. This was done using openDBcopy version 0.51rc2 (http://opendbcopy.sourceforge.net/). openDBcopy is an Open Source project by Anthony Smith, published under the terms of the GNU General Public License. It translates the schema read from any database (here the UKIDSSDR8PLUS release) into a new database (here a MySQL test database) using Hibernate to create intermediate XML files. During the translation the intermediate and new schema files were also checked for column names that were not compatible with either XML or MySQL reserved keywords.

The second step was data outgest into CSV files, with the columns reordered according to the new MySQL schema. This was done using the VDFS (VISTA Data Flow System) software.

Most of the preparation code was bundled into a Python wrapper script that is available on GitHub: https://github.com/lsst-uk/qservtestbed/tree/master/schema_translation.

After the CSV files and .sql files are outgested, some minor corrections are needed. Firstly, columns that were named as reserved key words in MySQL need to be renamed, such as 'dec' in the UKIDSS data release. Secondly, records with the default 'dec' or 'ra' values must be removed since they are designed only for the UKIDSS query requests and will be invalid as the sky coordinates for the spatial partition.

3.3.2 Data ingestion

Qserv treats non-partitioned and director/child tables differently. For director and child tables, Qserv calls the `sph-partition` command to partition the CSV files before copying them to the worker nodes. For non-partitioned tables, the partitioning step is skipped.

The `sph-partition` command takes the CSV file and partitioning configuration files as input, and outputs the text chunk files, which can then be used for ingestion. For convenience, partitioning parameters are usually specified in two configuration files, one for common parameters and one for each table. Here's an example of partitioning the udsDetection table with the `sph-partition` command:

```
sph-partition  --config-file=common.cfg \
--config-file=udsDetection.cfg --in.csv.null=NULL \
--in.csv.delimiter=$'\t' --in.csv.escape=\\ \
--in.csv.quote=\" --in=udsDetection.tsv  \
--mr.num-workers=6 --mr.pool-size=32768 \
--mr.block-size=16 --out.dir=chunks/udsDetection
```

Here are the configuration files for the `sph-partition` command:

```
# common.cfg
# Common partitioning parameters.
part = {
    num-stripes      = 22
    num-sub-stripes  = 7
    chunk            = chunkId
    sub-chunk        = subChunkId
    default-overlap  = 0.5
}

in.csv = {
    # input file format
    null      = '\\N'
    delimiter = '\t'
}

# Output CSV format.
out.csv = {
    null      = '\\N'
    delimiter = '\t'
    escape    = '\\'
    no-quote  = true
}
```

```
# udsDetection.cfg
# Source table primary key column.
id = multiframeID

# director table location
dirDb = UKIDSSDR8_udsDetection
dirTable = udsDetection

part = {
    pos = 'ra, decl'
    overlap = 0.5
}

in.csv = {
    # List of Source table column names, in order of
occurrence. Some fields are skipped here.
    field = [
      multiframeID
      extNum
      seqNum
      cuEventID
      filterID
      isoFlux
      isoMag
      ...
      ]
}
```

Several parameters are important here: the name of the sky coordinates columns (pos), and the chunk overlap radius (overlap) which is set to be 0.5 degree for all director tables in our experiment. The number of chunks, sub-chunks and overlap radius are important parameters and must be set with respect to users' query needs, since Qserv will not perform join queries across chunks. Also, the overlap radius must not be larger than the width of the chunk.

To be noted, the data partitioning is integrated with the `qserv-data-loader.py` script by default if the "`--skip-partition`" option is not specified, so it's usually not needed to run the `sph-partition` command manually.

After the tables are partitioned, the qserv-data-loader.py script could then be used to load the data. The following examples show how to load the non-partitioned tables, director tables and pre-partitioned tables (using `sph-partition`), respectively.

```
wmgr_options="--host=127.0.0.1 --port=5012 \
--secret=/path/to/wmgr.secret"
datapath="/path/to/data/"
workdir="/datapool/qserv_head/"

# use --one-table and --skip-partition options together
qserv-data-loader.py $wmgr_options -W WORKER1 \
--delete-tables --css-remove \
--empty-chunks=$workdir/var/lib/qserv/empty_qservTest.txt \
--config=$datapath/common.cfg \
--config=$datapath/udsSource.cfg \
--tmp-dir=/tmp/data-loader-tmp --one-table \
--skip-partition qservTest udsSource $datapath/udsSource.sql \
$datapath/udsSource.csv

# Partitioning parameters defined in .cfg files, which are the
# same as ones used by sph-partition
qserv-data-loader.py -v $wmgr_options -W WORKER1 \
--delete-tables --css-remove \
--empty-chunks=$workdir/var/lib/qserv/empty_qservTest.txt  \
--config=$datapath/common.cfg \
--config=$datapath/udsSource.cfg \
--tmp-dir=/tmp/data-loader-tmp \
--chunks-dir=/tmp/data-loader-chunks \
qservTest udsSource $datapath/udsSource.sql \
$datapath/udsSource.csv

# Files generated by sph-partition should be under /tmp/data-
# loader-chunks. csv not needed
qserv-data-loader.py $wmgr_options -W WORKER1 --delete-tables \
--css-remove \
--empty-chunks=$workdir/var/lib/qserv/empty_qservTest.txt \
--config=$datapath/common.cfg \
--config=$datapath/udsSource.cfg \
--chunks-dir=/tmp/data-loader-chunks --skip-partition \
qservTest udsSource $datapath/udsSource.sql
```

During ingestion, table information will be registered to the Central State System (CSS), and metadata table will be built on the head node. Chunks are loaded evenly to all worker nodes using a round-robin policy.

An empty-chunk file will be created at `/datapool/qserv_head/var/lib/qserv` for each database for optimisation of queries. The empty-chunk file is just a plain text file listing the numbers of the chunks that are empty.

3.3.3 Metadata registration

During the evaluation, Qserv had several issues which caused the join query between director tables to fail. In order to enable join queries, some complex "hack" must be done manually.

Firstly, the partition identifier of tables that can be joined must be set to be the same. The reason is that, Qserv requires tables to be equally partitioned if users try to perform join queries on them to avoid data exchange between worker nodes. And the way Qserv evaluates whether two tables are equally partitioned is by simply compare the partition identifiers of them.

This can be done by modifying the CSS data, which is stored in the qservCssData database on the master node. The following example shows how to set partition identifiers of the example databases (UKIDSSDR8_dxsSource and UKIDSSDR8_dxsDetection) to the same value (0000000001).

```
UPDATE                qservCssData.kvData                SET
kvVal='{"partitioningId":"0000000001","releaseStatus":"R
ELEASED","storageClass":"L2"}'                         WHERE
kvKey='/DBS/UKIDSSDR8_dxsSource/.packed.json';

UPDATE                qservCssData.kvData                SET
kvVal='{"partitioningId":"0000000001","releaseStatus":"R
ELEASED","storageClass":"L2"}'                         WHERE
kvKey='/DBS/UKIDSSDR8_dxsDetection/.packed.json';
```

Secondly, the chunks of tables that need to be joined must be "aligned" on all worker nodes. The reason for that is, the sph-partition tool names every chunk according to its position on the sky, so that every object close enough will end up with the same chunk number even if they are in different tables. One problem is that for some tables, the certain block of the sky can be empty and Qserv does not create empty chunk for it, at the time of writing. However, during join queries, Qserv still tries to find every corresponding chunk. So if one table holds a certain chunk while another does not, the query will crash. As a result, empty chunks must be manually created on worker nodes to enable join queries. This includes creating the chunk table, the overlap table, registering metadata on both the master node and worker nodes.

On the worker node, create chunk table and overlap table (with chunk number 444), and register them in the qservw_worker.Chunks table.

```
Create              table             UKIDSSDR8.Object_444
like UKIDSSDR8.Object_440;
Create         table        UKIDSSDR8.ObjectFullOverlap_444
like UKIDSSDR8.ObjectFullOverlap_440;
insert        into qservw_worker.Chunks        (db,chunk)
values("UKIDSSDR8",444);
```

On the master node, create metadata in the qservCssData.kvData table.

```
INSERT  INTO  qservCssData.kvData  (kvKey,  parentKvID)
value("/DBS/UKIDSSDR8/TABLES/Object/CHUNKS,1664);
INSERT  INTO  qservCssData.kvData  (kvKey,  parentKvID)
value("/DBS/UKIDSSDR8/TABLES/Object/CHUNKS/REPLICAS,1665
);
INSERT  INTO  qservCssData.kvData  (kvKey,  parentKvID)
value("/DBS/UKIDSSDR8/TABLES/Object/CHUNKS/REPLICAS/0000
000001,1666);
INSERT   INTO   qservCssData.kvData   (kvKey,   kvVal,
parentKvID)
value("/DBS/UKIDSSDR8/TABLES/Object/CHUNKS/REPLICAS/0000
000001/.packed.json,{"nodeName":"WORKER1"},1667);
```

For an installation with more than one worker node, things get more complicated because Qserv currently uses a simple round-robin policy to distribute all chunks, which means the chunks are by default randomly distributed. To avoid data exchange between worker nodes, all chunks with the same number must sit on the same node. To achieve that, some extra modification must be done in the qserv-data-loader.py script.

To make things easier, an ingesting tool, https://github.com/lsst-uk/qservtestbed/tree/master/ingest has been developed which handles all problems described in this section. This tool is a technology preview, which can help guide the ingestion process for an experienced Qserv administrator, but is not sufficiently robust for a non-expert.

4 Benchmarks

4.1 Aim

Having set up several versions of Qserv, and ingested example sky survey datasets ((UKIDSSDR8plus and BestDR7)), with which the DAC team are familiar, we next wanted to evaluate and benchmark Qserv using a suite of SQL queries that are indicative of likely LSST science use-case scenarios. Benchmarking was completed using the two node (head node plus one worker) configuration. The SQL queries were built around UKIDSS and SDSS database releases, and were known to run on an existing Microsoft SQL Server installation in the WSA.

The SQL queries are available from GitHub:

- https://github.com/lsst-uk/qservtestbed/tree/master/wsa — holds the original starting point for the queries working under MS SQL server
- https://github.com/lsst-uk/qservtestbed/tree/master/qserv — holds the re-worked queries for Qserv.

4.2 Query Sources

The queries used are collated from three sources:

- Queries named as RGM*.sql are inspired by science requirements outlined in the LSST:UK Phase B Preparation workshop, held at the Royal Observatory Edinburgh, 26th—28th April 2017. These took the form of brief statements of science goals for LSST data translated into roughly equivalent queries to run on the WSA.
- Queries named as NCH*.sql are inspired by two papers describing the WFCAM and VISTA Science Archives – [1] and [2].
- Queries named as DW*.sql are inspired by the LSST document page containing a compilation of expected LSST common queries. Queries are chosen based on query complication and common usage, designed to test the database capabilities – **Error! Reference source not found.**.

For each query, we reproduce the form of the query and discuss, in broad terms, our success in porting the query to work with Qserv and any performance differences observed.

4.3 Queries

RGM1.sql

A workshop attendee expressed the requirement of being able to perform forced photometry of VISTA images at the positions of interesting LSST sources. This query identifies the filenames of UKIDSS images around the positions of point sources selected according to a colour-magnitude cut intended to be find highly-redenned quasars.

```
select   mf.fileName,f.shortName,l.ra,l.decl,obstype,frameType   from
UKIDSSDR8_lasSource.lasSource as l,UKIDSSDR8.CurrentAstrometry as ca,
UKIDSSDR8.Multiframe                                                as
mf, UKIDSSDR8.MultiframeDetector as mfd, UKIDSSDR8.Filter as f where
mf.multiframeID=mfd.multiframeID  and  mf.multiframeID=ca.multiframeID
and
mfd.extNum=ca.extNum  and  f.filterID=mf.filterID  and  mf.fileName  !=
'NONE'
AND ((l.ra >= minRA and l.ra <= maxRA) or (l.ra + 360.0 >= minRA and
l.ra
+ 360.0 <= maxRA))  and ( l.decl >= minDec and l.decl <= maxDec)  and
obstype LIKE 'OBJECT' and frameType like '%stack' and j_1mhpnt +hmkpnt
>
2.5  and  kapermag3  >  0  and  kapermag3  <  16.5  AND  (priOrSec<=0  OR
priOrSec=frameSetID) AND yClass = -1.0 AND yppErrBits = 0 AND j_1Class
=
-1.0  AND  j_1ppErrBits  =  0  AND  hClass  =  -1.0  AND  hppErrBits  =  0  AND
(j_2Class=-1.0   OR   j_2Class   =   -9999)   AND   j_2ppErrBits   <=   0   AND
(kClass=-1.0 OR kClass = -9999)  AND kppErrBits <= 0 AND yXi BETWEEN
-1.0
AND 1.0 AND yEta BETWEEN -1.0 AND 1.0 AND j_1Xi BETWEEN -1.0 AND 1.0
AND
j_1Eta  BETWEEN  -1.0  AND  1.0  AND  hXi  BETWEEN  -1.0  AND  1.0  AND  hEta
BETWEEN
-1.0 AND 1.0 AND ((kXi BETWEEN -1.0 AND 1.0 AND kEta BETWEEN -1.0 AND
1.0)
OR kXi < -0.9e9);
```

A straightforward query joining one of the UKIDSS chunked tables with some metadata tables. The original query used a WSA view, `reliableLASpointsource`, not available under Qserv so the query was expanded to include the SQL of the view.

Issues: Initially there were issues with missing data in the Multiframe table and then in using "NOT LIKE" and -1 integers. Multiframe was re-loaded. The query was re-worked to use "LIKE" and -1.0. These bugs should now have been fixed by the developers.

Performance - the Qserv and MSSQL queries return the same rows. The MSSQL query is about 3 times quicker, some of this will be due to the indexes present in the MSSQL installation that are not present in Qserv.

RGM2.sql

A workshop attendee was interested in how to generate colour images of strong lens candidates. This query finds WSA filenames for Y, H and K images containing objects in a certain ellipticity range.

```
Select  s.ra  as  ra,s.decl  as  decl,y.fileName  as  yFile,  l.yeNum  as
yExtNum       from        UKIDSSDR8_lasSource.lasSource        as        s,
UKIDSSDR8.lasMergeLog      as      l,      UKIDSSDR8.Multiframe      as      y,
UKIDSSDR8.Multiframe      as      h,      UKIDSSDR8.Multiframe      as      k
where yEll between  0.8 and 0.85 and hEll between 0.8 and 0.85 and
kEll    between    0.8    and    0.85    and       s.frameSetID=l.frameSetID
and      y.multiframeID=ymfID      and      h.multiframeID=hmfID      and
k.multiframeID=kmfID;
```

A simple table join query. Using a chunked source table and metadata tables

Issues - none

Performance - same results, very similar times

RGM3.sql

A workshop attendee envisaged selecting sources with red centres and blue outer regions, to try and identify double source-plane gravitational lenses. This query implements something roughly analogous using pairs of aperture magnitudes.

```
select  ra,decl,frameSetID,yAperMag3,j_1AperMag3,hAperMag3,kAperMag3,
(hAperMag6-kAperMag6)-(hAperMag3-kAperMag6),
(j_1AperMag3-yAperMag3)-(j_1AperMag6-yAperMag6)
 from UKIDSSDR8_lasSource.lasSource as s where
yClass=1 and j_1Class=1 and hClass=1 and kClass=1 and yppErrBits=0 and
j_1ppErrBits=0
and hppErrBits=0 and kppErrBits=0
and (hAperMag6-kAperMag6)-(hAperMag3-kAperMag3) >1
and (j_1AperMag3-yAperMag3)-(j_1AperMag6-yAperMag6) > 1
and j_1AperMag3>0 and yAperMag3 >0 and j_1AperMag6 > 0 and yAperMag6
>0
and kAperMag3 > 0 and hAperMag3 >0 and kAperMag6 > 0 and hAperMag6 >
0
order by yAperMag3;
```

A simple single chunked source table scan.

Issues - could not assign aliases to arithmetic combination eg select A-B as C. Issue reported to devs

Performance - same results, very similar times

RGM4.sql

A workshop attendee was interested in comparing photometric redshifts estimated using different algorithms. SDSS DR8 contains two photo-z tables, but we were using DR7, as that has neighbour tables with UKIDSS DR8, so we had to join the photoz table with itself in this query.

```
select pz1.z as z1,pz1.zErr as zErr1, pz2.z as z2,pz2.zErr as zErr2
from BestDr7.Photoz as pz1, BestDr7.Photoz as pz2
where pz1.objID=pz2.objID and pz1.z > 0.3 and pz1.z < 0.4
select pz1.z, pz1.zerr, pz2.z, pz2.zerr from bestdr8..photoz as pz1,
bestdr8..photozrf as pz2 where pz1.objid=pz2.objid and pz1.z > 0.3 and
pz1.z < 0.4
```

An SDSS table join query. Large number of rows are returned, so ran from shell e.g.

mysql --host 127.0.0.1 --port 4040 --user qsmaster -e "select blah from blah;" > results.txt

Issues - we're using BestDr7 as that is neighboured with UKIDSSDR8. The table photozrf was introduced in BestDR8 so for now just join photoz with itself.

Performance – Qserv was 3—4 times slower than MS SQL Server. Possibly this is an issue in staging of results?

RGM5.sql

A workshop attendee was interested in extracting information from existing catalogues close to the position of a newly-discovered LSST transient, so this query extracts photometric and classification info from the WSA around a particular position.

```
select ra,decl,yDeblend,yAperMag3,yClass,yppErrBits,
j_1Deblend,j_1AperMag3,j_1Class,j_1ppErrBits,hDeblend,hAperMag3,hClas
s,hppErrBits,
kDeblend,kAperMag3,kClass,kppErrBits
from UKIDSSDR8_lasSource.lasSource where
qserv_areaspec_circle(181.6,-0.6,0.0333333);
```

Cone search of chunked table using `qserv_areaspec_circle`

Issues - none

Performance - same results, very similar times

RGM6.sql

A workshop attendees wanted to select stellar tracers of Galactic structure in (u-g,g-r) space, so this query attempts to select stars from the UKIDSS Large Area Survey (LAS) on the basis of their colours.

```
SELECT ra,decl,yAperMag3,j_1Apermag3,hAperMag3 FROM
UKIDSSDR8_lasSource.lasSource WHERE (priOrSec <= 0 OR priOrSec =
frameSetID)
AND yClass BETWEEN -2.0 AND -1.0 AND yppErrBits < 256 AND j_1Class
BETWEEN -2.0 AND -1.0 AND j_1ppErrBits < 256
AND hClass BETWEEN -2.0 AND -1.0 AND hppErrBits < 256 AND (j_2Class
BETWEEN -2.0 AND -1.0 OR j_2Class = -9999)
AND (j_2ppErrBits < 256) AND (kClass BETWEEN -2.0 AND -1.0 OR
kClass = -9999) AND (kppErrBits < 256)
AND yAperMag3-j_1Apermag3 < 0 AND abs(j_1AperMag3 - j_2AperMag3 ) <
0.05;
```

Another simple single chunked source table scan

Issues - Qserv didn't like where x = -1 even though the attributes were ints, changed to -1.0 etc. Reported to devs. Originally this query was to use a view of lassource but views not available(?) in Qserv so expanded SQL constraints to include the view selection.

Performance - same results, very similar times

RGM7.sql

A workshop attendee wanted to identify intrinsically faint white dwarf stars on the basis of high proper motions.

Issues - no proper motions available in UKIDSSDR8, not implementable in either system but would be a single table trawl.

RGM8.sql

A workshop attendee was interested in stellar lightcurves to probe stellar rotation periods, and this query models extraction of multi-epoch photometry from the UKIDSS Deep Extragalactic Survey (DXS).

```
SELECT m.mjdObs,d.aperMag3,d.aperMag3Err,d.ppErrBits,d.seqNum,x.flag
FROM UKIDSSDR8_dxsDetection.dxsDetection AS d,
UKIDSSDR8.dxsSourceXDetectionBestMatch AS x, UKIDSSDR8.Multiframe AS
m WHERE x.sourceID=446677639289 AND x.multiframeID=d.multiframeID
AND x.extNum=d.extNum AND x.seqNum=d.seqNum AND
x.multiframeID=m.multiframeID AND d.filterID=5 ORDER BY mjdObs;
```

Table join query.

Issues - Tables must be specified in the correct order in the query: Need to put the chunked table, UKIDSSDR8_dxsDetection.dxsDetection, first to get it to work. Otherwise crashes Qserv.

Performance - same results, very similar times

RGM9.sql

A workshop attendee wanted to discover candidate exoplanet microlensing events as sources that brighten by more than three magnitudes in thirty days. This query looks for highly-variable sources in DXS.

```
SELECT x.sourceID AS xid,min(d.aperMag3),max(d.aperMag3),count(*) FROM
UKIDSSDR8_dxsSource.dxsSource                    AS                     s,
UKIDSSDR8.dxsSourceXDetectionBestMatch AS x, UKIDSSDR8.dxsVariability
AS       v,       UKIDSSDR8_dxsDetection.dxsDetection       as        d
WHERE     s.sourceID=v.sourceID     AND     s.mergedClass=-1.0     AND
v.variableClass=1       and       abs(kMinMag-kMaxMag)       >3       and
x.sourceID=s.sourceID
and    x.multiframeID=d.multiframeID    AND    x.extNum=d.extNum    AND
x.seqNum=d.seqNum and scisql_angSep(s.ra, s.decl, d.ra, d.decl) < 0.01
and d.aperMag3 > 0 GROUP BY xid;
```

Table joins including joins between two chunked tables, source & detection

Issues - Needed to expand sub-query as not available in Qserv. Then Qserv cannot make use of the neighbour table and the query needs a `scisql_angSep` to make it run. Reported to devs, neighbour table use is planned.

Performance – much, much slower, lots of trigonometric calculations.

RGM10.sql - (query below from MS SQl Server)

A workshop attendee was interested in finding candidate high-z quasars that are present in VISTA, but not in LSST. This query attempted to model that through selecting UKIDSS LAS sources that do not have neighbours in the SDSS.

```
select               ra                ,dec,              framesetid,
yapermag3,j_1apermag3,hapermag3,kapermag3,mergedclass  from  lassource
as                                                                   s
where  ra  between  325  and  355  and  dec  between  -1  and  -0.5
and sourceid not in (select masterobjid from lasSourceXDR7PhotoObjAll
)
and     yapermag3>0     and     j_1apermag3>0     and     hapermag3     >0
and kapermag3>0 and mergedclass=-1
```

Uses NOT IN sub-query

Issues - can't think how to implement in Qserv

NCH1.sql - query not currently working, got this far

This query was to identify the most flaring stars, ordered by number of candidate flares.

```
SELECT v.sourceID as vID, s.ra as sra, s.decl as sdecl, v.framesetID
as
vFSID, knGoodObs as gknGoodObs, kMinMag as gkMinMag, kmedianMag as
gkmedianMag, kMaxMag as gkMaxMag, variableClass as gvariableClass,
mergedClass  as  gmergedClass,  count(*)  as  nBrightDetections  FROM
UKIDSSDR8_dxsSource.dxsSource                    AS                  s,
UKIDSSDR8.dxsSourceXDetectionBestMatch
AS          b,          UKIDSSDR8.dxsVariability          AS          v,
UKIDSSDR8_dxsDetection.dxsDetection                                  as
d      WHERE   s.sourceID=v.sourceID   AND   b.sourceID=v.sourceID   AND
b.multiframeID=d.multiframeID         AND      b.extNum=d.extNum     AND
b.seqNum=d.seqNum
and scisql_angSep(s.ra, s.decl, d.ra, d.decl) < 0.01 and kmedianMag<19.
and
kmedianMag>0.   AND   knGoodObs>=5   AND   kbestAper=5   AND   (kmedianMag-
kMinMag)>0.5
AND   kMinMag>0.   AND   d.seqNum>0   AND   d.ppErrBits   IN   (0,16)   AND
d.filterID=5                                                         AND
d.aperMag5>0 AND d.aperMag5<(kmedianMag-0.5) group by vID,  sra, sdecl,
vFSID,  gknGoodObs,  gkMinMag,  gkmedianMag,  gkMaxMag,  gvariableClass,
gmergedClass;
```

Table joins that should make use of the neighbour table

Issues - as with RGM9 need to add scisql_angSep to get this to parse so not able to make efficient use of the neighbour table. Then hit a problem with group'ing by multiple attributes. Reported to devs, probably fixed but we were not running latest version.

Performance - not working but using scisql_angSep will be much slower.

NCH2.sql

This query was to extract lightcurves for the above flaring stars.

MS SQL query reads:

```
select                          s.sourceID,d.filterID,mjdObs,aperMag3
from  dxsDetection  d,  dxsSource  s,  dxsSourceXDetectionBestMatch  x,
Multiframe                                                          m
where s.sourceID = x.sourceID and x.multiframeID = d.multiframeID and
x.extNum                          =                          d.extNum
and  x.seqNum  =  d.seqNum  and  m.multiframeID  =  x.multiframeID  and
d.aperMag3                                >                         0
and                 s.sourceID                  in                  (
SELECT                                                     v.sourceID
FROM dxsVariability as v, dxsSource as s, dxsSourceXDetectionBestMatch
as                                                                 b,
dxsDetection                            as                          d
WHERE     s.sourceID=v.sourceID     AND     b.sourceID=v.sourceID     AND
b.multiframeID=d.multiframeID
AND        b.extNum=d.extNum        AND        b.seqNum=d.seqNum        AND
/* select the magnitude range, brighter than Ks=17 and not default. */
kMedianMag<19.             and             kMedianMag>0.             AND
/*           at           least           5          observations        */
knGoodObs>=5              AND              kBestAper=5                AND
/* Min mag is at least 2 magnitudes brighter than median mag(but minMag
is                    not                    default)                   */
(kMedianMag-kMinMag)>0.5         AND         kMinMag>0.          AND
/* Only good K band detections in same aperture as statistics are
calculated                                                          in*/
d.seqNum>0 AND d.ppErrBits IN (0,16) AND d.filterID=5 AND d.aperMag5>0
AND                                         d.aperMag5<(kMedianMag-0.5)
/*                 Group                    detections                  */
GROUP         BY          v.sourceID,         s.ra,          s.dec,
v.framesetID, knGoodObs, kMinMag, kMedianMag, kMaxMag, variableClass,
mergedClass
HAVING                                                       COUNT(*)>2
)
order by s.sourceID,d.filterID,mjdObs
```

Issues – not implementable, as uses subquery, and cannot work out how to re-express it otherwise.

NCH3.sql - not implementable, uses CAST.

This query genrates counts-in-cells of sources in the UKIDSS Galactic Plane Survey (GPS).

MS SQL query reads:

```
SELECT      CAST(ROUND(l*6.0,0)      AS      INT)/6.0      AS      lon,
CAST(ROUND(b*6.0,0)         AS         INT)/6.0         AS         lat,
COUNT(*)                         AS                                 num
FROM                                                          gpsSource
WHERE       k_1Class       BETWEEN       -2       AND       -1       AND
k_1ppErrBits                  <                  256                 AND
/*       Make       a       seamless       selection       (i.e.       exclude
duplicates)       in       any       overlap       regions:       */
(priOrSec=0                   OR                   priOrSec=frameSetID)
/*    Bin    up    in    10    arcmin    x    10    arcmin    cells:    */
GROUP         BY         CAST(ROUND(l*6.0,0)         AS         INT)/6.0,
CAST(ROUND(b*6.0,0) AS INT)/6.0
```

Issues – not implementable, as uses CAST, and cannot work out how to re-express it otherwise.

NCH4.sql

This query was to select quasar candidates on the basis of optical/near-infrared colours.

MS SQL query reads:

```
SELECT              psfMag_i-psfMag_z              AS              imz,
psfMag_z-j_1AperMag3                       AS                       zmj,
psfMag_i-yAperMag3                         AS                       imy,
ymj_1Pnt                             AS                             ymj
FROM               lasPointSource               AS               s,
lasSourceXDR7PhotoObj                       AS                       x,
BestDR7..PhotoObj                          AS                         p
WHERE
/*                 Join                    predicates:                  */
s.sourceID               =               x.masterObjID               AND
x.slaveObjID               =               p.objID               AND
x.distanceMins               <               1.0/60.0               AND
/*       Select       only       the       nearest       primary       SDSS
point               source               crossmatch:               */
x.distanceMins                           IN                           (
SELECT                                              MIN(distanceMins)
FROM                                             lasSourceXDR7PhotoObj
WHERE         masterObjID          =          x.masterObjID         AND
sdssPrimary                  =                  1                  AND
sdssType                              =                             6
)                                                                   AND
/*       Remove        any        default        SDSS       mags:     */
psfMag_i                     >                  0.0                AND
/*       Colour        cuts       for       high-z       QSOs       from
Hewett          et          al.          (2006)         and     Venemans
et                    al.                    (2007):                  */
psfMag_i-yAperMag3                >                 4.0            AND
ymj_1Pnt                      <                  0.8               AND
psfMagErr_u                      >                  0.3            AND
psfMagErr_g                      >                  0.3            AND
psfMagErr_r > 0.3
```

Issues – not implementable, as uses subquery, and cannot work out how to re-express it otherwise.

NCH5.sql

This query selects extragalactic variable sources.

```
SELECT
s.sourceID,s.ra,s.decl,v.frameSetID,v.jmedianMag,v.jMagRms,v.jnGoodOb
s,v.jskewness,v.hmedianMag,
v.hMagRms,v.hnGoodObs,v.hskewness,                       v.kmedianMag,
v.kMagRms,v.knGoodObs,v.kskewness,
jMaxMag-jMinMag,hMaxMag-hMinMag,kMaxMag-kMinMag
FROM          UKIDSSDR8_dxsSource.dxsSource          AS            s,
UKIDSSDR8.dxsVariability  AS  v   WHERE  v.sourceID=s.sourceID  AND
s.mergedClass      IN      (-1.0,-2.0)      AND      v.variableClass=1
AND  (((jMaxMag-jMinMag)>0.1  AND  jMinMag>0.  AND  jnGoodObs>=5)  OR
((hMaxMag-hMinMag)>0.1     AND     hMinMag>0.     AND     hnGoodObs>=5)
OR  ((kMaxMag-kMinMag)>0.1 AND kMinMag>0. AND knGoodObs>=5));
```

Table join query.

Issues - as previously can't assign column name to arithmetic combination of attributes. Reported to devs.

Performance - similar times, one more result row (rounding differences).

DW1.sql

This query (based on query 070) was to perform multiple joins between tables.

Issues - not currently implementable, as requires subquery function

DW2.sql

This query (based on query 007) selects time series data for sources in a given area of sky in a given photometric band and with a given variability index.

```
SELECT s.sourceID as ssID, v.jmeanMag as meanMag, m.mjdObs as MJD,

       d.aperMag3 as obsMag, d.aperMag3err as errMAg
```

```
FROM          UKIDSSDR8_dxsDetection.dxsDetection          AS          d,
UKIDSSDR8_dxsSource.dxsSource                       AS                   s,
UKIDSSDR8.dxsSourceXDetectionBestMatch AS x,

UKIDSSDR8.Multiframe AS m,

UKIDSSDR8.dxsVariability AS v

WHERE x.sourceID=s.sourceID

AND   x.multiframeID=d.multiframeID

AND   x.extNum=d.extNum

AND   x.seqNum=d.seqNum

AND   x.multiframeID=m.multiframeID

AND   d.filterID=3

AND   v.sourceID=s.sourceID

AND   v.jprobVar > 0.8

AND   d.aperMag3 > -100

AND   scisql_angSep(d.ra, d.decl,334.25,0.3) < 0.5

AND   scisql_angSep(d.ra, d.decl,s.ra,s.decl) < 0.1

ORDER BY ssID, MJD
```

Issues – Same with RGM8.sql above. Tables cannot be in any order in the query. Need to put the UKIDSSDR8_dxsDetection.dxsDetection table first for query to work.

Performance - same results, big difference in timing. WSA: 37 seconds, QServ 2hrs6mins. The WSA MS SQL query is not having to do any trig to join source and detection tables.

DW3.sql

This query (based on query 047) selects variable objects near galaxies.

```
SELECT v.sourceID as vID, v.ra as vRA, v.decl as vDec, g.sourceID as dID, g.ra as dRA, g.decl as dDec, scisql_angSep(g.ra, g.decl, v.ra, v.decl) as angsep

FROM UKIDSSDR8_dxsSource.dxsSource AS g,

     UKIDSSDR8.dxsVariability AS v

WHERE v.sourceID <> g.sourceID

AND   g.pGalaxy > 0.99

AND   v.jprobVar > 0.9

AND   v.ra > 0.0

AND   scisql_angSep(g.ra, g.decl, v.ra, v.decl) < 0.00167
```

Issues – None, simple transfer

Performance – Some difference. Times out on WSA after 10,800 seconds (3 hours) but runs in 1hr 23mins on Qserv.

DW4.sql

This query (based on query 013) finds close pairs of sources with similar colours.

```
SELECT DISTINCT s1.sourceID AS ID_1, s2.sourceID AS ID_2

FROM UKIDSSDR8_dxsSource.dxsSource as s1,

     UKIDSSDR8_dxsSource.dxsSource as s2

WHERE scisql_angSep(s1.ra, s1.decl, s2.ra, s2.decl) < 0.0167

AND   s1.sourceID <> s2.sourceID

AND   ABS( s1.jmhPnt - s2.jmhPnt ) < 0.5

AND   ABS( s1.hmkPnt - s2.hmkPnt ) < 0.5

AND   ABS( s1.jmkPnt - s2.jmkPnt ) < 0.5
```

Issues – None, simple transfer to Qserv

Performance – Big difference. Times out on WSA after 10,800 seconds (3 hours) but runs in 9mins 35secs on Qserv[3].

DW5.sql

This query (based on query 029) was to find the magnitude of the closest source within a given distance of a particular location.

Issues - Not currently implementable, as requires subquery function

DW6.sql

This query (based on query 025) was to find all galaxies in regions of high source density.

Issues - Not currently implementable, as requires subquery function

DW7.sql

This query (based on query 043) is to find all stellar neighbours within a certain distance for which at least one of the pair has the colours of a white dwarf.

```
SELECT  S1.objID AS WD, S2.objID AS Star
FROM  BestDR7_PhotoObjAll.PhotoObjAll S1,
  BestDR7_PhotoObjAll.PhotoObjAll S2
WHERE S1.type =  6
AND   S2.type =  6
AND   scisql_angSep(S1.ra, S1.decl, S2.ra, S2.decl)*3600 < 10
AND   S1.u-S1.g < 0.4
AND   S1.g-S1.r < 0.7
AND   S1.r-S1.i > 0.4
AND   S1.i-S1.z > 0.4
AND   S1.objID <> S2.objID
AND   scisql_angSep(S1.ra, S1.decl, S2.ra, S2.decl)<0.1
```

Issues – None, simple transfer to QServ

Performance – Times out on WSA, returns error from QServ.

DW8.sql

This query (based on query 034) is searching for merging galaxy pairs.

```
SELECT g1.objID AS Gal1_ID, g2.objID AS Gal2_ID
FROM BestDR7_PhotoObjAll.PhotoObjAll g1,
     BestDR7_PhotoObjAll.PhotoObjAll g2,
     BestDr7.Photoz n
WHERE g1.type = 3 AND g2.type = 3
AND   g1.objID = n.objID
AND   g2.objID = n.nnObjID
AND   g1.objId < g2.ObjID
AND   g1.petrorad_u > 0 AND g2.petrorad_u > 0
AND   g1.petrorad_g > 0 AND g2.petrorad_g > 0
AND   g1.petrorad_r > 0 AND g2.petrorad_r > 0
AND   g1.petrorad_i > 0 AND g2.petrorad_i > 0
AND   g1.petrorad_z > 0 AND g2.petrorad_z > 0
AND   g1.petroradErr_g > 0 AND g2.petroradErr_g > 0
AND   g1.petroMag_g BETWEEN 16 AND 21
```

```
    AND    g2.petroMag_g BETWEEN 16 AND 21
    AND    g1.modelmag_u > -9999
    AND    g1.modelmag_g > -9999
    AND    g1.modelmag_r > -9999
    AND    g1.modelmag_i > -9999
    AND    g1.modelmag_z > -9999
    AND    g2.modelmag_u > -9999
    AND    g2.modelmag_g > -9999
    AND    g2.modelmag_r > -9999
    AND    g2.modelmag_i > -9999
    AND    g2.modelmag_z > -9999
    AND    abs(g1.modelmag_g - g2.modelmag_g) > 3
    AND    (g1.petroR50_r BETWEEN 0.25*g2.petroR50_r AND 4.0*g2.petroR50_r)
    AND    (g2.petroR50_r BETWEEN 0.25*g1.petroR50_r AND 4.0*g1.petroR50_r)
    AND        (scisql_angSep(g1.ra,   g1.decl,   g2.ra,   g2.decl)/3600.0   <=
                                    (g1.petroR50_r + g2.petroR50_r))
    AND    scisql_angSep(g1.ra, g1.decl, g2.ra, g2.decl) < 0.1
```

Issues – None, simple transfer to QServ

Performance – Times out on WSA, runs in 5hrs 01mins 10secs. Returns only one result.

DW9.sql

This query (based on query 048) was to identify variable sources in clusters

Issues – Not currently implementable, uses subquery.

DW10.sql

This query (based on query 066) was to identify extremely red objects.

Issues – Not currently implementable, uses many subqueries.

4.4 Other queries

As well as the "science" queries above, we also considered a comparison of table trawls and index searches.

Table scan

```
    select * from gpsSource where mergedClassStat=-111.94543301;
```

Qserv took about 10 minutes, about twice the time of MSSQL

Index lookup

```
    select * from gpsSource where jApermag3=7.12345;
```

There is an index on jApermag3 in MSSQL so as expected 10 minutes in Qserv (as it's doing a full scan) vs 1 sec in MSSQL

Using the primary key of director table

```
select max(sourceID) from gpsSource;
```

Qserv - 2 sec

```
select * from gpsSource where sourceID=438916075061
```

Qserv - 1 sec

As expected Qserv is fast at these lookups.

4.5 Conclusions

Coming from a background of freeform case-insensitive SQL queries, it took a while to get used to writing Qserv SQL. Ignoring hardware differences, overall performance (full table trawls) is similar. Any minor issues with syntax parsing and bugs were quickly fixed by the LSST DM Team.

As expected, not being able to use subqueries is an unresolved issue, as is not being able to make proper use of the neighbour tables. It is possible that some of the above queries that use subqueries could be re-written but we could not readily see a solution. Also one needs to investigate if deploying views is possible. Some queries need to have tables in a certain order.

Having more than one director table makes loading/distributing the data more complex. Generally, the Qserv installation is a bit “flaky” queries that were working, stop working and Qserv has to be re-started. Often the error reporting is not that useful and you have to go hunting through the logs.

5 References

[1] Hambly et al., “The WFCAM Science Archive”, 2008, MNRAS

[2] Cross et al., “The VISTA Science Archive”, 2012, A&A

[3] LSST Common Queries Source: https://dev.lsstcorp.org/trac/wiki/db/queries.