

Sequoia

Administrator's Guide

Release 5.0

Table of Contents

Preface	iii
1. Installation	1
1.1. Download	1
1.2. Software requirements	1
1.2.1. Synchronize node clocks using Network Time Protocol (NTP)	1
1.3. Network requirements	2
1.4. Install	2
1.5. Configuration	3
1.5.1. Configure the first controller	3
1.5.2. Configure the second controller	4
1.5.3. Configure JDBC Client	5
1.6. Automatic startup	5
1.7. Activate two controllers	6
2. Upgrade	8
2.1. Download	8
2.2. Upgrade	8
3. Downgrade	9
3.1. Downgrade	9
4. Disaster Recovery	10
4.1. Recover from a controller failure	10
4.2. Recover from a database server failure	11

Preface

Sequoia is a transparent middleware solution for offering clustering, load balancing and failover services for any database. Sequoia is the continuation of the C-JDBC project#. The database is distributed and replicated among several nodes and Sequoia balances the queries among these nodes. Sequoia handles node failures and provides support for checkpointing and hot recovery.

Here is a list of Sequoia features and requirements :

- No modification of existing applications or databases.
- Operational with any database providing a JDBC driver.
- High availability provided by advanced RAIDb technology.
- Transparent failover and recovery capabilities.
- Performance scalability with unique load balancing and query result caching features.
- Integrated JMX-based administration and monitoring.
- 100% Java implementation allowing portability across platforms with a JRE 1.4 or greater.
- Open source licensed under Apache v2 license.
- Professional support, training and consulting provided by Continuent Inc.

The Sequoia web site [<http://sequoia.continuent.org>] carries details on the latest release and other information to make your work or play with PostgreSQL more productive. And The mailing lists are a good place to have your questions answered, to share experiences with other users, and to contact the developers.

1

Installation

1.1. Download

Download and save Sequoia binary archive from Continuent Forge [<http://forge.continuent.org>].

```
cd /usr/local
```

1.2. Software requirements

1.2.1. Synchronize node clocks using Network Time Protocol (NTP)

Edit the NTP configuration file on each node Next, edit the NTP configuration file `ntp.conf` on each node to synchronize them and ensure that the node clocks have the same time value. The `ntp.conf` file is read when the NTP daemon is started. By default this file can be found at `etc/ntp.conf` . To configure `ntp.conf` on the controller nodes

Use the `restrict` keyword to restrict access to the NTP services in the cluster.

```
restrict default kod notrap nomodify nopeer noquery  
restrict 127.127.1.0 nomodify  
restrict <network ip address> mask <network mask>
```

Use `localhost` as the synchronization source.

```
server 127.127.1.0
```

Configure a reference clock

```
fudge 127.127.1.0 stratum 5
```

List the other node(s) in the cluster as peer NTP servers. Do not include the node whose configuration you are editing in the list of peer servers.

```
peer <IP address>
```

Configure NTP to create a drift file in the directory `var/lib/ntp` .

```
driftfile /var/lib/ntp/ntp.drift
```

Example 1.1. The NTP configuration file of node1 with IP address 192.168.1.8 is, as follows

```
restrict default kod notrap nomodify nopeer noquery
restrict 127.127.1.0 nomodify
restrict <network ip address> mask <network mask> driftfile /var/lib/ntp/ntp.drift
```

Example 1.2. The NTP configuration file of node2 with IP address 192.168.1.9 is, as follows:

```
restrict default kod notrap nomodify nopeer noquery
restrict 127.0.0.1 nomodify
restrict 192.168.1.0 mask 255.255.255.0
server 127.127.1.0
fudge 127.127.1.0 stratum 5
peer 192.168.1.8
driftfile /var/lib/ntp/ntp.drift
```

1.3. Network requirements

Sequoia requires a network supporting TCP/IP for communication between cluster nodes. Use a switched Ethernet network: the recommendation is to use a full-duplex 1Gb/s interconnection between controllers and database servers.

In addition, the following proper network settings are required for the controller group communication to work.

If your controller host uses multiple IP addresses, make sure that your hostname translates to the real IP address of the controller node, and not the localhost address (127.0.0.1). To verify this, use the command `ping hostname` to output the real IP address of the host.

Define a default route for the network adapter bound by JGroups (usually eth0). If such a route does not exist, either the controller group communication initialization will fail or the controllers will be unable to see each other.

- To check for the default entries in your routing table under Linux, you can use the `/sbin/route` command.
- To add the default route, use a command such as `/sbin/route add default eth0`.

The use of Dynamic Host Configuration Protocol (DHCP) is strongly discouraged: use fixed IP addresses instead. It has been reported that the use of DHCP can either block (under Windows) or fail to properly set a default route.

1.4. Install

The archive name is « `sequoia-3.XX-bin.tar.gz` », « `XX` » is the minor version number. Install binaries and create required user and directories.

```

tar xzf sequoia-3.XX-bin.tar.gz
ln -s sequoia-3.XX-bin sequoia
mkdir -p /etc/sequoia/conf
mkdir -p /var/log/sequoia
mkdir -p /var/sequoia/dump
cd /usr/local/sequoia
cp -r config/* /etc/sequoia/conf
rm -rf config
ln -s /etc/sequoia/conf config
rm -rf ./log
ln -s /var/log/sequoia log
groupadd sequoiagroup
useradd -g sequoiagroup -s /bin/bash -d /home/sequoia -m sequoia
chown -R sequoia:sequoiagroup /usr/local/sequoia*
chown -R sequoia:sequoiagroup /etc/sequoia
chown -R sequoia:sequoiagroup /var/log/sequoia
chown -R sequoia:sequoiagroup /var/sequoia/dump
su - sequoia
echo "export JAVA_HOME=/usr/local/java">>~/.bash_profile
echo "export JVM_OPTIONS="-Xmx200m -Xms200m">>~/.bash_profile
echo "export SEQUOIA_HOME=/usr/local/sequoia">>~/.bash_profile
source ~/.bash_profile
echo $JAVA_HOME
echo $JVM_OPTIONS
echo $SEQUOIA_HOME

```

1.5. Configuration

We propose to configure two clustered controllers with same virtual database (logical clustered database) in order to provide fail-over mechanism. And each controller must be support several backends in different RaidDB mode. In default configuration, we use raidDB1 mechanism to mirror all databases backends hosted by a controller. We use PostgreSQL database, but we can use another database.

1.5.1. Configure the first controller

Edit controller configuration /etc/sequoia/conf/controller/controller.xml

```

</SEQUOIA-CONTROLLER>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE SEQUOIA-CONTROLLER PUBLIC "-//Continuent//DTD SEQUOIA-CONTROLLER 3.XX//EN"
"http://sequoia.continuent.org/dtds/sequoia-controller-3.XX.dtd">
<SEQUOIA-CONTROLLER>
  <Controller name="Controller1" jdbcIpAddress="0.0.0.0" jdbcPort="25322">
    <JmxSettings jmxIpAddress="<network address>" jmxPort="1090"/>
    <VirtualDatabaseAutoLoad configFile="postgres-raidb1-distribution-1.xml"
      virtualDatabaseName="<logical name of database cluster>" autoEnableBackends="true"/>
  </Controller>
</SEQUOIA-CONTROLLER>

```

Configure virtual database /etc/sequoia/conf/virtualdatabase/postgres-raidb1-distribution-1.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SEQUOIA PUBLIC "-//Continuent//DTD SEQUOIA 3.XX//EN"
"http://sequoia.continuent.org/dtds/sequoia-3.XX.dtd">
<SEQUOIA>
  <VirtualDatabase name="<logical name of database cluster>" maxNbOfConnections="1000">
    <Distribution>

```

```

        <MessageTimeouts/>
    </Distribution>
    <Backup>
        <Backuper backuperName="pgdump"
            className="org.continuent.sequoia.controller.backup.backupers.PostgreSQLBinaryBackuper" />
    </Backup>
    <AuthenticationManager>
        <AdminUser username="admin" password="" />
        <VirtualUser vLogin="user" vPassword="" />
    </AuthenticationManager>
    <DatabaseBackend name="<database node name>" driver="org.postgresql.Driver"
        url="jdbc:postgresql://<database network address>/<database name>"
        connectionTestStatement="select now()">
        <ConnectionManager vLogin="user" rLogin="postgres" rPassword="">
            <VariablePoolConnectionManager initPoolSize="10" minPoolSize="5"
                maxPoolSize="50" idleTimeout="30" waitTimeout="10" />
        </ConnectionManager>
    </DatabaseBackend>
    <DatabaseBackend name="<database node name>" driver="org.postgresql.Driver"
        url="jdbc:postgresql://<database network address>/<database name>"
        connectionTestStatement="select now()">
        <ConnectionManager vLogin="user" rLogin="postgres" rPassword="">
            <VariablePoolConnectionManager initPoolSize="10" minPoolSize="5"
                maxPoolSize="50" idleTimeout="30" waitTimeout="10" />
        </ConnectionManager>
    </DatabaseBackend>
    ...

    <DatabaseSchema/>
    <RequestManager>
        <RequestScheduler>
            <RAIDb-1Scheduler level="passThrough" />
        </RequestScheduler>
        <RequestCache>
            <MetadataCache/>
            <ParsingCache/>
            <ResultCache granularity="database">
                <DefaultResultCacheRule timestampResolution="36000">
                    <EagerCaching/>
                </DefaultResultCacheRule>
            </ResultCache>
        </RequestCache>
        <LoadBalancer>
            <RAIDb-1>
                <WaitForCompletion policy="first" />
                <RAIDb-1-LeastPendingRequestsFirst/>
            </RAIDb-1>
        </LoadBalancer>
        <RecoveryLog driver="org.postgresql.Driver"
            url="jdbc:postgresql://localhost/recovery" login="postgres" password="">
            <RecoveryLogTable/>
            <CheckpointTable/>
            <BackendTable/>
            <DumpTable/>
        </RecoveryLog>
    </RequestManager>
</VirtualDatabase>
</SEQUOIA>

```

Add required JDBC postgresQL driver in `/usr/local/sequoia/drivers`, you can download here [<http://jdbc.postgresql.org>].

1.5.2. Configure the second controller

It's same as first, define controller, virtual database and add required JDBC driver.

1.5.3. Configure JDBC Client

Add the Sequoia JDBC library in your classpath, you can found sequoia-driver.jar file in /usr/local/sequoia/drivers/. Use the driver class : org.continuent.sequoia.driver.Driver and this url :

```
jdbc:sequoia://<first controller network address>,<second controller network address>/<logical name of da
```

1.6. Automatic startup

Edir default startup script in standard directory, /etc/init.d/sequoia file.

```
#!/bin/sh
#
# Sequoia start/stop script for database cluster controller.
#
DAEMON=/usr/local/sequoia/bin/controller.sh
NAME="sequoia"
DESC="Sequoia Database cluster controller"

test -x $DAEMON || exit 0
case "$1" in
    start)
        ps -U sequoia | grep -q java
        if [ $? -eq 0 ] ; then
            echo "Sequoia is already running"
            echo "Please, stop it first"
            exit 1
        else
            echo "Starting $DESC: $NAME"
            su - sequoia -c "${DAEMON}" & >/dev/null 2>/dev/null
            echo "."
        fi
        ;;
    stop)
        echo "Stopping $DESC: $NAME "

# FIXME : add cleanly stop
#
# sequoia@lmCBS-db01 (lm-at-crm-db01) /usr/local/sequoia/bin:$ ./console.sh
#
# Launching the Sequoia controller console
#
# Initializing Controller module...
#
# Initializing VirtualDatabase Administration module...
#
# Initializing SQL Console module...
#
# Sequoia driver (Sequoia core v3.10) successfully loaded.
#
# localhost:1090 > show virtualdatabases
#
# myDB
#
# localhost:1090 > shutdown virtualdatabase myDB
#
# Virtual database Administrator Login > admin
#
# Virtual database Administrator Password >
#
# Shutting down database myDB from controller
#
# Database myDB was successfully shutdown
#
# localhost:1090 > shutdown
#
# Shutdown was complete
#
# localhost:1090 > Exiting Sequoia controller console
#
sleep 15
```

```

ps -U sequoia | grep -q java
if [ $? -eq 0 ] ; then
    ps h -U sequoia | grep java | awk '{print $1}' |xargs kill
    sleep 2
    echo "Sequoia didn't stoped after 15 seconds, has been killed"
    sleep 2
    ps -U sequoia | grep -q java
    if [ $? -eq 0 ] ; then
        ps h -U sequoia | grep java | awk '{print $1}' |xargs kill -9
        sleep 2
        echo "WARNING ! ! Sequoia couldn't be killed, has been hardly killed (kill -9) !"
    fi
fi
echo "."
;;
*)
echo "Usage: /etc/init.d/$NAME {start|stop}" >&2
exit 1
;;
esac

exit 0

```

Add execution right and register automatic startup.

```

chmod u+x /etc/init.d/sequoia
update-rc.d -f sequoia defaults

```

1.7. Activate two controllers

Run these operation on two controllers.

Start the controller.

```

/etc/init.d/sequoia start

```

Connect to administration console of controller.

```

su - sequoia
cd /usr/local/sequoia/bin
./console.sh
Initializing Controller module...
Initializing VirtualDatabase Administration module...
Initializing SQL Console module...
Sequoia driver (Sequoia core v3.XX) successfully loaded.
localhost:1090 > admin <logical name of database cluster>
Virtual database Administrator Login > admin
Virtual database Administrator Password >
Ready to administrate virtual database <logical name of database cluster>

```

For each backends hosted by current controller, enable backend :

```

<logical name of database cluster>(admin) > initialize <database node name>
Virtual Database <logical name of database cluster> has been successfully initialized from backend <data
<logical name of database cluster>(admin) > enable <database node name>
Enabling backend <database node name> from its last known checkpoint

```

2

Upgrade

2.1. Download

Same as installation section.

2.2. Upgrade

Upgrade old version to the new version.

```
/etc/init.d/sequoia stop
cd /usr/local
tar xzf sequoia-3.XX-bin.tar.gz
rm /usr/local/sequoia
ln -s sequoia-3.XX-bin sequoia
cp -r /etc/sequoia/conf /etc/sequoia/conf.YYYYMMDD
cd /usr/local/sequoia
rm -rf config
ln -s /etc/sequoia/conf config
rm -rf ./log
ln -s /var/log/sequoia log
chown -R sequoia:sequoiagroup /usr/local/sequoia*
chown -R sequoia:sequoiagroup /etc/sequoia
/etc/init.d/sequoia start
```

3

Downgrade

3.1. Downgrade

Downgrade the current version (3.XX) to the previous (3.YY).

```
/etc/init.d/sequoia stop
rm /usr/local/sequoia
ln -s /usr/local/sequoia-3.YY-bin /usr/local/sequoia
rm /etc/sequoia/conf/*
cp -r /etc/sequoia/conf.YYYYMMDD /etc/sequoia/conf
/etc/init.d/sequoia start
```

4

Disaster Recovery

4.1. Recover from a controller failure

Start the failed controller and start the command line client.

```
/etc/init.d/sequoia start
su - sequoia
cd $SEQUOIA_HOME/bin
./console.sh
```

Connect to the second controller and go to admin mode, and start the command line client.

```
su - sequoia
cd $SEQUOIA_HOME/bin
./console.sh
localhost:1090 > admin <logical name of database cluster>
localhost:1090 > show backends
<logical name of database cluster>(admin)> backup <one backend hosted by this controller> nodel.dump pgdu
Backend login for backup process > postgres
Backend password for backup process >
Backup backend nodel in dump file nodel.dump
```

Check your dump is available in list of databased dump files.

```
show dumps
```

Copy the database dump to the failed controller.

```
<logical name of database cluster>(admin)> transfer dump nodel.dump <failed controller network address>:1
Transferring dump nodel.dump to controller XXX.XXX.XXX.XXX:1090...
Done
```

Go to admin mode and enable the use of the expert admin commands.

```
expert on
```

Synchronize the recovery log of the failed controller.

```
restore log nodel.dump <failed controller network address>:1090
```

Disable the use of the expert admin commands.

```
expert off
```

Recover the second : On the failed controller, go to admin mode.

```
localhost:1090 > admin <logical name of database cluster>
Virtual database Administrator Login > admin
Virtual database Administrator Password >
Ready to administrate virtual database <logical name of database cluster>
<logical name of database cluster>(admin) > restore log node1.dump <good controller network address>:1090
<logical name of database cluster>(admin) > restore backend node2 node1.dump
Backend login for restore process > postgres
Backend password for restore process >
Restoring backend node2 with dump node1.dump
```

Enable the database server of the failed controller. Enabling the server synchronizes it with the other cluster nodes.

```
<logical name of database cluster>(admin) > enable node2
Enabling backend node2 from its last known checkpoint
```

If you are using a configuration where there are more than one database servers per controller, repeat the recover operation for the other database server(s) to activate it.

4.2. Recover from a database server failure

To activate a database server after a database server failure, for example, you must first restore a dump file to it and then enable it. If you do not have a dump file that you can use to restore the database, create one as instructed in [Back up a database server](#).

Go to admin mode on the controller that hosts the database server you want to activate.

```
su - sequoia
cd $SEQUOIA_HOME/bin
./console.sh
localhost:1090 > admin <logical name of database cluster>
Virtual database Administrator Login > admin
Virtual database Administrator Password >
Ready to administrate virtual database <logical name of database cluster>
<logical name of database cluster>(admin) >
```

Show the available database dump files.

```
show dumps
```

Use a database dump to restore the database server that you want to activate.

```
restore backend <backend name> <dump name>
```

Enable the database server. Enabling the server synchronizes it with the other cluster nodes.

```
enable <backend name>
```