

Reliable DUCC - Design

Written and maintained by the Apache
UIMATMDevelopment Community

Copyright © 2012 The Apache Software Foundation

Copyright © 2012 International Business Machines Corporation

License and Disclaimer The ASF licenses this documentation to you under the Apache License, Version 2.0 (the "License"); you may not use this documentation except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, this documentation and its contents are distributed under the License on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Trademarks All terms mentioned in the text that are known to be trademarks or service marks have been appropriately capitalized. Use of such terms in this book should not be regarded as affecting the validity of the the trademark or service mark.

Publication date: April 2019

Multiple DUCC head nodes

This first major section describes support for multiple active DUCC head nodes.

Introduction

DUCC can be configured to run reliably by having multiple head nodes, comprising one *master* and one or more *backup* head nodes. DUCC exploits Linux *keepalived* virtual IP addressing to enable this capability.

The advantages are that if the *master* DUCC host becomes unusable, the *backup* DUCC can take over seamlessly such that active distributed Jobs, Reservations, Managed Reservations and Services continue uninterrupted. Take over also facilitates continued acceptance of new submissions and monitoring of new and existing submissions without interruption.

Daemons

Each head node, whether *master* or *backup*, runs a Broker, Orchestrator, PM, RM, and SM.

The Cassandra database is expected to be located on a node(s) separate from the head nodes.

Likewise, the JD node(s) is separate from the head nodes.

The Agents are distributed, as before.

Configuring Host Machines

See *Configuring Simple Virtual IP Address Failover Using Keepalived* which can be found at this web address: https://docs.oracle.com/cd/E37670_01/E41138/html/section_uxg_lzh_nr.html.

Sample MASTER /etc/keepalived/keepalived.conf

```
! Configuration File for keepalived
```

```
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.6.253
    }
}
```

Sample BACKUP /etc/keepalived/keepalived.conf

```
! Configuration File for keepalived
```

```
vrrp_instance VI_1 {
    state BACKUP
    interface eth0
```

```

    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        192.168.6.253
    }
}

```

Linux Commands

Starting keepalived

```

> sudo service keepalived start
Starting keepalived: [ OK ]

```

Querying keepalived

```

> /sbin/ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:21:5e:20:02:84 brd ff:ff:ff:ff:ff:ff
    inet 192.168.3.7/16 brd 192.168.255.255 scope global eth0
    inet 192.168.6.253/32 scope global eth0
    inet6 fe80::221:5eff:fe20:284/64 scope link
        valid_lft forever preferred_lft forever

```

Stopping keepalived

```

> sudo service keepalived stop
Stopping keepalived:

```

Configuring DUCC

To configure DUCC to run reliable, one required property must be configured in the *site.ducc.properties* file. Example:

```
ducc.head = 192.168.6.253
```

Use the virtual IP address configured for your host machines keepalived. Use of the DNS name is also supported.

Webserver

Webserver for Master

The *master* DUCC Webserver will display all pages normally with additional information in the heading upper left:
reliable: **master**

Webserver for Backup

The *backup* DUCC Webserver will display some pages normally with additional information in the heading upper left:

reliable: **backup**

Hovering over reliable will yield the following information: *Click to visit master*

Several pages will display the following information (or similar):

```
no data - not master
```

Database

Configure the database to be on a separate machine from the reliable DUCC head nodes. In *site.ducc.properties* specify:

```
# Database location
ducc.database.host = dbhost123
ducc.database.jmx.host = dbhost123
ducc.database.automanager = false
```

The existing administrator commands *start_ducc* and *stop_ducc* will honor the value specified for *ducc.database.automanager* in *site.ducc.properties*.

Code changes

The key changes include a new script (see *ducc_head_mode.py*) to interact with Linux to determine virtual IP address status and corresponding Java code (see *common.head.ADuccHead.java*) that interprets the status to make transitions between *master* and *backup* states.

new scripts

ducc_head_mode.py

This is a new script employed at runtime by the various daemons to determine the current mode of operation. Status is determined through invocation of this script upon receipt of each Orchestrator publication.

```
# purpose:      determine reliable ducc status
# input:        none
# output:       one of { unspecified, master, backup }
# operation:    look in ducc.properties for relevant keywords
#              and employ linux commands to determine if system
#              has matching configured virtual IP address
```

existing and new scripts

ducc_post_install - no need to create webserver request log directory

ducc_util.py - incorporate host name into *cassandra.pid*, *cassandra.console* path; broker host must be local host; head node must be eligible with respect to *keepalived.conf*; head node local components are all daemons except Database and Agents; fix remote db stop; honor *ducc.database.automanager* flag in *site.ducc.properties*

ducc.py - incorporate host name into *cassandra.pid*, *cassandra.console* path

start_ducc.py - head node local components must on eligible local host

start_sim - (example) honor database autostart flag in *ducc.properties* item **stop_sim** - (example) honor database autostart flag in *ducc.properties*

configuration files

ducc.properties

```
# The name of the node where DUCC runs.
# This property declares the node where the DUCC administrative processes run (Orchestrator,
# Resource Manager, Process Manager, Service Manager). This property is required and MUST be
# configured in new installation. The installation script ducc_post_install initializes this
# property to the node the script is executed on.
# Reliable DUCC: if running reliably, then this value must be the same as that specified
# for the virtual_ipaddress in /etc/keepalived/keepalived.conf. DUCC CLI and Agents employ
# this value to connect to the current reliable DUCC head node.
ducc.head = <head-node>
```

Although not strictly true, the Orchestrator, RM, SM, PM, Webserver and Broker "must" all be configured on the head node. Reliable DUCC may work with other configurations, but it has not been tested as such.

```
# If set to true, DUCC will start and stop the Cassandra database as part of its normal
# start/stop scripting.
ducc.database.automanage = true
```

log4j.xml

```
Add DUCC\_NODENAME to log file name for OR, RM, PM, SM, and system-events.
This allows reliable DUCC head nodes to share the same ducc\_runtime directory
in the filesystem without collisions.
```

agent

DuccWorkHelper - use virtual IP address configured as ducc.head node

AgentEventListener - ignore any incoming publications from backup producer

CGroupsTest - employ changed DuccIdFactory signature

ServiceTester - broker must be on ducc.head node

cli

DuccMonitor - use WS node or virtual IP address configured as ducc.head node

DuccUiUtilities - use virtual IP address configured as ducc.head node (to submit, cancel..)

common

AbstractDuccComponent - remove commented-out code, remove print to console, head node local components are all daemons except Database and Agents

ADuccHead - abstract class with reliable DUCC share functionality

IDuccHead - reliable DUCC interface

IDuccEnv - remove DUCC.LOGS.WEBSERVER.DIR, not used

IStateServices - database access control RW or RO

NullStateServices - database access control RW or RO

StateServices - database access control RW or RO

DuccDaemonRuntimeProperties - incorporate hostname into logs directory location
InetHelper - incorporate hostname into logs directory location
DuccPropertiesHelper - fetch virtual IP address configured as ducc.head node
DuccPropertiesResolver - Remove key ducc.broker.hostname, broker must be on ducc.head node
IDuccLoggerComponents - Missing PM abbreviation
DuccIdFactory - improved (generalized) to handle DB persisted sequence numbering

database

IDuccHead - reliable DUCC interface
DbOrchestratorProperties - support for OR properties table
IDbOrchestratorProperties - interface of OR properties table
IOrchestratorProperties - interface for OR properties
IOrchestratorProperties - database access control RW or RO

orchestrator

DuccHead - loggable wrapper around common.ADuccHead
OrchestratorCommonArea - add restart capability for transition to *master*
OrchestratorComponent - reject requests from CLI and JD and publications for Agents when not *master*, employ DB for Job and publication number persistence, use active workMap from common area, tag publication with node identity and producer state *master* or *backup*, make transitions between *master* and *backup* states
OrchestratorRecovery - employ changed DuccIdFactory initialization requirements
ReservationFactory - employ changed DuccIdFactory signature
StateJobAccounting - log job state changes
StateManager - use active workMap from common area
WorkMapHelper - adding logging
AOrchestratorCheckpoint - refactor checkpointing, suspend when *backup* resume when *master*
IOrchestratorCheckpoint - refactor checkpointing
OrchestratorCheckpoint - refactor checkpointing
OrchestratorCheckpointDb - refactor checkpointing
OrchestratorCheckpointFile - refactor checkpointing
OrchestratorConfiguration - employ changed DuccIdFactory for publication sequence numbering
OrDbDuccWorks - specification to DB only when *master*
OrDbDuccWorks - orchestrator properties to DB only when *master*
OrchestratorEventListener - record to system events log daemon switches between *backup* and *master*
ReservationFactory - employ changed DuccIdFactory for Job numbering
ReservationFactory - employ changed DuccIdFactory signature
JdScheduler - suspend JD host management when *backup* resume when *master*
HealthMonitor - use active workMap from common area

MaintenanceThread - use active workMap from common area
AOrchestratorState - refactor orchestrator state managements from files to DB
DuccWorkIdFactory - refactor orchestrator state managements from files to DB
IOrchestratorState - refactor orchestrator state managements from files to DB
OrchestratorState - refactor orchestrator state managements from files to DB
OrchestratorStateDb - refactor orchestrator state managements from files to DB
OrchestratorStateDbConversion - refactor orchestrator state managements from files to DB
OrchestratorStateFile - refactor orchestrator state managements from files to DB
AOrchestratorStateJson - refactor orchestrator state managements from files to DB
SystemEventsLogger - record all CLI interactions in system events log
TestSuite - print whether *backup* or *master*

pm

DuccHead - loggable wrapper around common.ADuccHead
ProcessManagerComponent - make transitions between *master* and *backup* states

sm

DuccHead - loggable wrapper around common.ADuccHead
ServiceHandler - resume operations when state is *master*, quiesce operations when state is *backup*
ServiceManagerComponent - make transitions between *master* and *backup* states, reject requests when in *backup* state, employ changed DuccIdFactory signature **ServiceSet** - handle new state *Dispossessed*

transport

JobDriverStateExchanger - use virtual IP address configured as ducc.head node
AbstractDuccEvent - tag publications with producer host identity and state *master* or *backup*
DaemonDuccEvent - switch to *master* or *backup* state for recording to system event log
DuccEvent - add events SWITCH_TO_MASTER and SWITCH_TO_BACKUP
JdEvent - interrogate publications producer state *master* or *backup*
IService - add service state Dispossessed, Service is not controlled by this Service Manager

webserver

BrokerHelper - use local host name to find co-located broker
DuccBoot - make boot reusable for switch to *master*
DuccData - create reset function for switch to *master*
DuccHead - loggable wrapper around common.ADuccHead
WebServerComponent - make transitions between *master* and *backup* states; incorporate hostname info logs directory location

WebServerConfiguration - make boot reusable for switch to *master*

DuccHandler - servlet to produce reliable DUCC state *master* or *backup*

DuccHandlerClassic - servlets to produce no data - not master when appropriate

DuccHandlerJsonFormat - servlets to produce no data - not master when appropriate

DuccWebServer - add method getPort; use host as part of request log directory path; incorporate hostname info logs directory location

DuccWebServerHelper - incorporate hostname info logs directory location

c4-ducc-mon.jsp - web page header location for reliable DUCC state

ducc.js - web page header updating for reliable DUCC state

examples

start_sim - broker must be on head node

Installing and Cloning

This second major section describes support for installation of head node master and backup(s).

TBD

Autostart

This third major section describes support for autostart of head node and agent daemons.

TBD

Monitoring and Switching

This fourth major section describes support monitoring of multiple head nodes and switching to an alternate when the primary is dysfunctional.

TBD