Developer`s Overview of Sonic

Praveen Chaudhary
Sr. Software Engineer
LinkedIn
Developer`s Overview of Sonic

To Become a Sonic Developer

1.) https://github.com/Azure/SONiC/wiki
2.) User Guide
3.) Developer`s overview Session(s).
   ** New Sonic Wiki will have link to User Guide & OCP.
   ** https://sonicswitch.slack.com/messages (Join Slack)

Praveen Chaudhary
LinkedIn
pchaudhary@linkedin.com
1. Redis-server, different DBs, view of key-value set, basic libraries to interact with Redis.

2. Components interaction.


4. Orchagent [Swss Docker]

5. Make file overview to build a new Feature with Sonic Base image or in a new docker.
<table>
<thead>
<tr>
<th>DB name</th>
<th>DB No.</th>
<th>Description</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPL_DB (Application DB)</td>
<td>0</td>
<td>ARP/NDP Entries, BGP Routes, LLDP entries, Next-Hop etc.</td>
<td>ROUTE_TABLE: INTF_TABLE: NEIGH_TABLE: VLAN_MEMBER_TABLE: PORT_TABLE: COPP_TABLE: LLDP_ENTRY_TABLE:</td>
</tr>
</tbody>
</table>
| ASIC_DB                 | 1      | Running ASIC Configuration and ASIC State Data. | "ASIC_STATE:SAI_OBJECT_TYPE_ROUTE_ENTRY:{"dest":"xxxx:f3g5:60:4::12b/128","switch_id":"oid:0x27770000000000"}
                          | | | "ASIC_STATE:SAI_OBJECT_TYPE_NEXT_HOP_GROUP_MEMBER:oid:0x2d000000003363" |
                          |        |                         | "ASIC_STATE:SAI_OBJECT_TYPE_NEIGHBOR_ENTRY:{"ip":"xxxx:f4x7:470:a750::2/126","rif":"oid:0x600000000000","switch_id":"oid:0x21000000000000"}" |
| COUNTERS_DB             | 2      | Counter data for port, lag, queue, ACLs. | COUNTERS: CRM:ACL_STATS:INGRESS:LAG: |

https://github.com/Azure/sonic-swss-common/blob/master/common/schema.h

| LOGLEVEL_DB             | 3      | Log level control swssloglevel -p for Sonic subsystems | buffermgrd NOTICE
                          | | | fpmsyncd NOTICE |
                          | | | intfsyncd NOTICE |
                          | | | intfsyncd NOTICE |
                          | | | neighsyncd NOTICE |
                          | | | orchagent NOTICE |
                          | | | portsyncd NOTICE |
                          | | | syncd NOTICE |
                          | | | teamsyncd NOTICE |
                          | | | vlannmgrd NOTICE |

Augmented Backus-Naur Form (ABNF) RFC 5234
Redis DB: Key-Value data & Python libraries

- "PORT|Ethernet4"
  1) "alias"
  2) "Eth2/1"
  3) "lanes"
  4) "69,70"
  5) "description"
  6) "xxQ|switch-in-dc.nw|Ethernet112"
  7) "fec"
  8) "xx"

- "ROUTE_TABLE:xxxx:f547:4551:21b::/64"
  1) "ifname"
  2) "Ethernet64,Ethernet66,Ethernet68,Ethernet70,Ethernet72,Ethernet74,Ethernet76,Ethernet78"
  3) "nexthop"
  4) "xxxx:f547:40:4027::1,xxxx:f547:40:4067::1,xxxx:f547:40:40a7::1,xxxx:f547:40:40e7::1,xxxx:f547:40:4127::1,xxxx:f547:40:4167::1,xxxx:f547:40:41a7::1,xxxx:f547:40:41e7::1"

- "ASIC_STATE:SAI_OBJECT_TYPE_ROUTE_ENTRY":{"dest":"xx.aaa.239.128/26","switch_id":"oid:0x21000000000000","vr":"oid:0x3000000000042"}
  1) "SAI_ROUTE_ENTRY_ATTR_NEXT_HOP_ID"
  2) "oid:0x50000000000000000000000000000000"

Code Path for Python Libraries:

Redis DB: interact using Python libraries

Main Classes:

```python
class DBInterface(object):
    REDIS_HOST = '127.0.0.1'
    REDIS_PORT = 6379
    REDIS_UNIX_SOCKET_PATH = '/var/run/redis/redis.sock'
    CONNECT_RETRY_WAIT_TIME = 10
    DATA_RETRIEVAL_WAIT_TIME = 3
    PUB_SUB_NOTIFICATION_TIMEOUT = 10.0  # seconds
    PUB_SUB_MAXIMUM_DATA_WAIT = 60.0  # seconds
    KEYSPACE_PATTERN = '__key*__:*'
    KEYSPACE_EVENTS = 'KEA'
    db_map = dict()

    def __init__(self, **kwargs):
        pass

    def db_list(self):
        pass

    def get_dbid(cls, db_name):
        pass

    def connect(self, db_name, retry_on=True):
        pass

    def onetime_connect(self, db_name):
        pass

    def persistent_connect(self, db_name):
        pass

    def subscribe_keyspace_notification(self, db_name):
        pass

    def get_redis_client(self, db_name):
        pass

    def publish(self, db_name, channel, message):
        pass

    def exists(self, db_name, key):
        pass

    def keys(self, db_name, pattern='*):
        pass

    def get(self, db_name, _hash, key):
        pass

    def get_all(self, db_name, _hash):
        pass

    def set(self, db_name, _hash, key, val):
        pass
```

```python
class SonicV2Connector(DBInterface):

class ConfigDBConnector(SonicV2Connector):
```

Example Code:

Example1: Fetch all keys from VLAN table from Config DB.
```python
kwargs = {}
if redis_unix_socket_path:
    kwargs['unix_socket_path'] = redis_unix_socket_path
config_db = ConfigDBConnector(**kwargs)
config_db.connect(wait_for_init=False)
data = config_db.get_table('VLAN')
keys = data.keys()
```

Example2: Fetch set of key-value pair for a bvid from ASIC DB:
```python
db = SonicV2Connector(**redis_kwargs)
db.connect('ASIC_DB')
vlan_entry = db.get_all('ASIC_DB', vlan_obj[0], blocking=True)
vlan_id = vlan_entry[b'SAI_VLAN_ATTR_VLAN_ID']
```

Sample Output:
```
Sample Output: Yes
```
Redis DB: interact with C++ libraries

C++ Libraries:

https://github.com/Azure/sonic-swss-common/blob/master/common/dbconnector.cpp
https://github.com/Azure/sonic-swss-common/blob/master/common/table.cpp

Example Code:

Access all <keys> and <key-value> pair from PORT_TABLE of CONFIG_DB using C++ libs:

```
DBConnector cfgDb(CONFIG_DB, DBConnector::DEFAULT_UNIXSOCKET, 0);
Table table(&cfgDb, CFG_PORT_TABLE_NAME, CONFIGDB_TABLE_NAME_SEPARATOR);
std::vector<FieldValueTuple> values;
std::vector<string> keys; table.getKeys(keys);
for (auto &k : keys )
{
   table.get(k, ovalues);
   /---- My Code ----/
}
```
Orchagent Processes:

APPL_DB
ASIC_DB
CONFIG_DB
STATE_DB

Code Path:
https://github.com/Azure/sonic-swss/blob/master/orchagent/orchdaemon.cpp

```c
void RouteOrch::doTask(Consumer & consumer) {
    auto it = consumer.m_toSync.begin();
    while (it != consumer.m_toSync.end()) {
        /* Process as per the role */
        Route_orch::func() [SAI call()]
        /* Erase it, if success. */
        it = consumer.m_toSync.erase(it);
        continue;
    }
}
```

SAI: status = sai_route_api->create_route_entry(&unicast_route_entry, 1, &attr);
https://github.com/Azure/sonic-swss/tree/201803/orchagent
class IntfMgr : public Orch
{
    public:
        IntfMgr(DBConnector *cfgDb, DBConnector *appDb, DBConnector *stateDb, const vector<string> &tableNames);
    using Orch::doTask;
    private:
        ProducerStateTable m_appIntfTableProducer;
        Table m_cfgIntfTable, m_cfgVlanIntfTable;
        Table m_statePortTable, m_stateLagTable, m_stateVlanTable;
        bool setIntfIp(const string &alias, const string &ipPrefixStr);
        void doTask(Consumer &consumer);
        void isIntfStateOk(const string &alias);
};
List of open source code repo used in Sonic:

FRR & Zebra: https://github.com/FRRouting/frr

LLDP: https://github.com/vincentbernat/lldpd.git

LLDPMGRD: https://github.com/Azure/sonic-buildimage/blob/master/dockers/driver-lldp-lldpmgrd

SNMP: https://sourceforge.net/projects/net-snmp/files/net-snmp/5.7.3/

Sonic_snmp agent: https://github.com/Azure/sonic-snmpagent/tree/master/src


Dhcp_Relay: https://salsa.debian.org/berni/isc-dhcp.git

CLI: https://github.com/Azure/sonic-utilities
Sonic Makefile Overview:

New Daemon Package including Makefile or .DSC

sonic-buildimage src/<newd>/Makefile

sonic-buildimage/rules/<newd>.mk

sonic-buildimage/slave.mk

sonic-buildimage/Makefile

MAIN_TARGET = $(NEWD)
DERIVEDTargets = $(LIBNEWD)
$(addprefix $(DEST)/, $(MAIN_TARGET)): $(DEST)/% :
  rm -rf ./newd
  # Clone newd repo
  git clone https://github.com/awesomeDev/newd.git newd
  git push newd
  # Build source and debian packages
  dpkg-buildpackage -ufakeroot -b -us -uc -j$(SONIC_CONFIG_MAKE_JOBS)
popd
  # Move the newly-built .deb packages to the destination directory
  mv $(NEWD)_$(LIBNEWD) $(DEST)/
$(addprefix $(DEST)/, $(DERIVED_targets)): $(DEST)/% : $(DEST)/$(MAIN_TARGET)

# newd package
NEWD_VERSION = 1.0.1
NEWD = newd $(NEWD_VERSION)-0_amd64.deb
$(NEWD)DEPENDS += $(LIBDEP)
$(NEWD)RDEPENDS += $(LIBRDEP)
$(NEWD)SRC_PATH = $(SRC_PATH)/newd
SONIC_MAKE DEBS += $(NEWD)
LIBNEWD = libnewd $(NEWD_VERSION)-0_amd64.deb
$(eval $(call add_derived_package,$(NEWD),$LIBNEWD))

# Export these variables so they can be used in a sub-make
export NEWD_VERSION
export NEWD
export LIBNEWD
Sonic Makefile Overview:

New Daemon Package including Makefile or .DSC

```bash
# Install new daemon version 1.0.1 with sonic base image.
sudo dpkg --set-selections >> /etc/apt/preferences.d/sonic-debian

# Install new daemon version 1.0.1 with sonic base image.

# Install new daemon version 1.0.1 with sonic base image.
sudo dpkg --set-selections >> /etc/apt/preferences.d/sonic-debian

# Install new daemon version 1.0.1 with sonic base image.
```

Add Package with Host Image

```bash
### files/build_templates/sonic_debian_extension.j2
Add Package with Host Image

```
Summary:

To Become a Sonic Developer
1.) https://github.com/Azure/SONiC/wiki
2.) User Guide
3.) Developer’s overview Session(s).