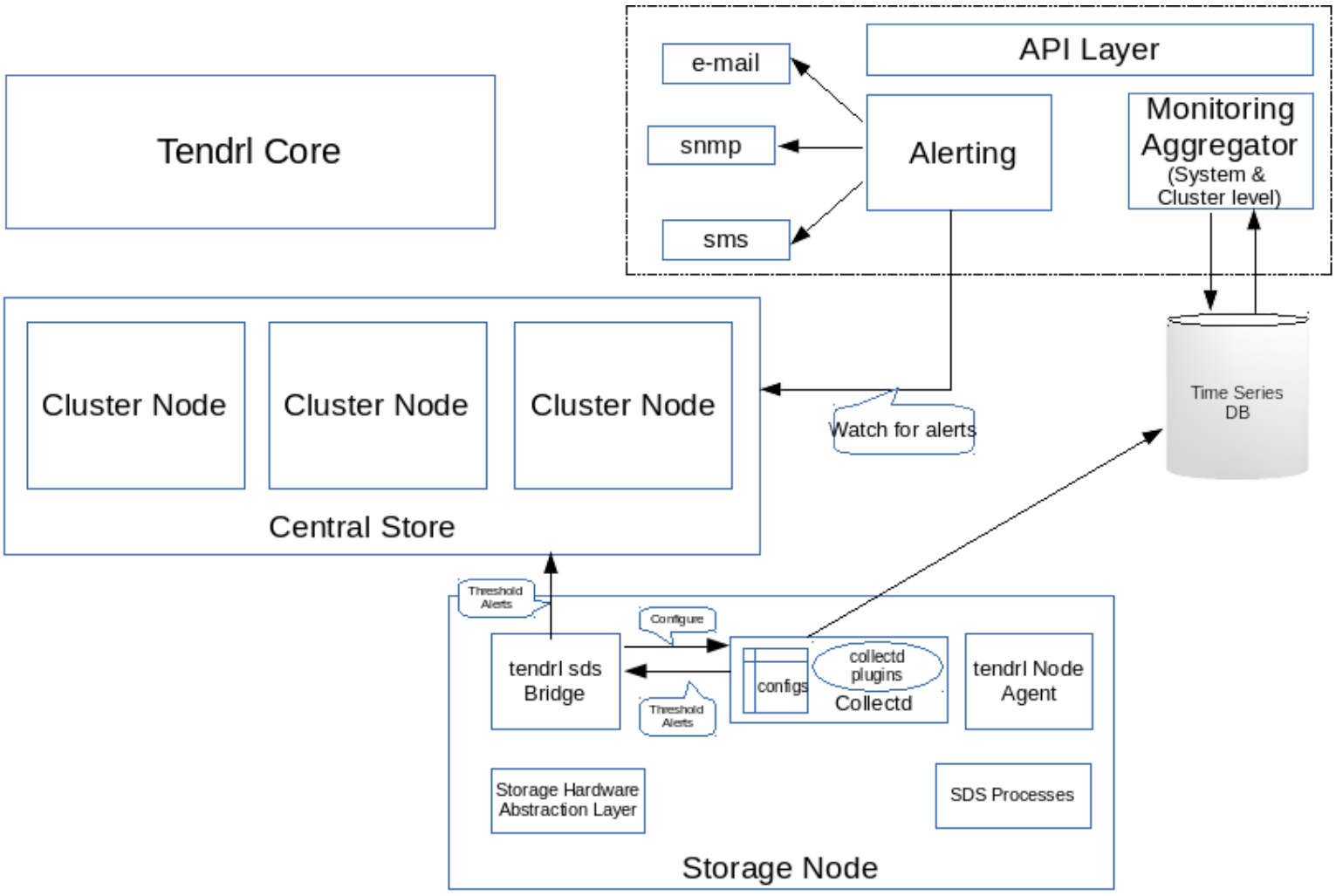
MONITORING ARCHITECTURE PROPOSAL



STACK COMPONENTS:

The monitoring stack will consist of the following parts:

1. Central Monitoring Application: It is a separate service that runs outside tendrl core stack and it relies on the state information in tendrl's central store for its functioning.It consists of the following parts
    a. Api Layer : This layer will serve the following responsibilities
        i. Serve the time series data stored in time series database.
        ii. Alert subscription management for users in tendrl core(This functionality will be achieved for users in tendrl core. These subscriptions will be stored in tendrl's central store).
    b. Aggregator : This module aggregates atomic stats(ex: cluster utilization) to composite(system utilization) stats which will be required by UI for the dashboards.
    c. Alerting : This module will serve the following responsibilities
        i. Send out alerts to destinations configured using Api Layer.
        ii. Watch for new alert/notification details in central store every configured intervals of time.
2. Time Series DB : This will hold the data collected by collectd and also those aggregated by the aggregator.
3. Collectd: This component resides on every node managed by tendrl. It serves the following responsibilities
    a. Collect the physical and logical resource utilizations using the configured plugins.
    b. Watch for the breach of configured thresholds and trigger execution of custom plugin.(The plugin sends the threshold breach information to the sds-bridge over a socket that is exposed by it).
    c. Push the stats collected to the configured write destination using the plugin configured as write plugin(For example, the write_graphite plugin will push stats to the graphite db).

PACKAGING:

1. Monitoring components on nodes are packaged into 2 categories:
    a. Sds specific :
        i. The collectd plugins under this category are designated to measure/collect the utilizations of the sds specific resources and hence accordingly, if tendrl supports n sds systems, there will be essentially n different plugin packages.
        ii. Apart from the collectd plugins, the templates for configuring these plugins will also be maintained as part of these packages.
        iii. Ex: gluster_monitoring, ceph_monitoring
        iv. Typically these packages will be installed on a subset of the nodes in the sds-cluster(ex: Mon nodes in ceph's case and the whole set of nodes in gluster's case)
    b. Physical resource specific :
        i. Plugins under this category can be classified as under:
            1. The plugins designated to measure/collect the utilizations of the resources generic to node like cpu, memory, swap, network, etc… These plugins come readily packaged with collectd
            2. Plugin to post the threshold breach information to sds-bridge over a socket exposed by it.
        ii. Configuration templates to configure the above plugins
        iii. This package will be installed on all the nodes.
2. Monitoring application:
    a. This will pull in time series db as a dependency.
    b. It will be usually installed on the node where tendrl is installed but nothing enforces this condition as the communication from this application to the tendrl app if any will be via rest apis.

NOTE:

1. collectd plugins will be loaded and configured only through collectd configuration files. Mere presence of plugins will not suffice for the plugins to be run and hence respective data to be collected.
2. Different configuration files for each plugin and then including them through a path wild card in the main config file is possible and used in Skyring 2.0
3. The configurations of a particular plugin will include its corresponding threshold configuration if applicable and required.
4. For monitoring the logical resources, it suffices to have collectd monitor for stats on:
    - In gluster's case, any random node in the cluster.
    - In ceph's case, any mon node and we can particularly choose it to be ceph leader mon.

Collectd Plugin Configuration Management:

Skyring 2.0 used salt-stack's formula based configuration generation capabilities. However, python's jinja templating can be an effective and powerful way to generate configuration files from jinja templates. A POC for the same has been tried out as in:

1. https://github.com/Tendrl/bridge_common/pull/27 -- Conf file generator
2. https://github.com/Tendrl/ceph_bridge/pull/11 -- Template

Generic flow:

1. As the final step of creating/managing an entity in tendrl, the appropriate collectd plugin configurations will be made on suitable nodes using the approach as described in section "**Collectd Plugin Configuration Management"** above.
2. After the step 1, the configured plugins start collecting the respective utilizations at every configured intervals of time and push them to the write destination using the plugin configured as write plugin(ex: write_graphite collectd plugin if configured as write plugin in collectd pushes data collected by collectd to the graphite db).
3. Parallely, every configured intervals of time the aggregator module of the central monitoring application will collect the instant value of stats from the time series db and update the stats of interest to tendrl into the tendrl's central store and also aggregates these stats @ cluster and system levels push back these stats to time series db and also to the central store. These aggregations are typically inline with the UX designs/requirements.
4. If thresholds configured are breached for any particular resource, a corresponding notification is generated in the collectd by the collectd's threshold plugin with all required details like resource name, the plugin name for which threshold breach was detected, the threshold value, the current value, etc… The plugin configured to be executed on Notification detected in collectd, will then be invoked internally by the collectd with all the above parameters. This plugin can then pass on these notification/threshold breach parameters to the sds-bridge(chosen as it contains all the sds specific intelligence that will be required for alert correlations) which will then update the suitable states in central store.
5. The Alerting module of the central monitoring application will keep a watch on the states in the central store every configured intervals of time and send out appropriate alerts to appropriate destinations.
6. Any queries for time series database will be served by the monitoring application's api layer and not the tendrl core.

Example Flows:

Manage node:

1. As a final step of this procedure, the configurations corresponding to physical resources plugins will be generated on the node.
2. Collectd starts collecting statistics for the utilizations of the configured physical resources and pushes them to time series db.
3. If for any resource, for which the thresholds are configured; the thresholds are breached, the threshold handling plugin will post the threshold breach details to the sds-bridge which will then make appropriate changes in the central store.
4. The alerting module of the central monitoring application which is observing the tendrl's central store for state changes related to threshold breaches, will then send out the alerts as configured.

Create Cluster:

1. As a final step of cluster creation, configuration is added on the appropriate node(and the node on which this configuration was made is noted in the central store) to the colletcd's config directory to run a plugin to gather cluster's utilization with an entry similar to:

   <Plugin "exec">

   Exec "skyring-user" "/usr/lib64/collectd/cluster_utilization.py <cluster_name>"

   </Plugin>

2. Collectd starts collecting cluster utilization and pushes them to time series db.
3. If cluster utilization crosses configured thresholds, the threshold handling plugin will post the threshold breach details to the sds-bridge which will then make appropriate changes in the central store.
4. The alerting module of the central monitoring application which is observing the tendrl's central store for state changes related to threshold breaches, will then send out the alerts as configured.

5. If the node on which the configurations are made is observed to have gone down, an alternative node is chosen by the …..

API formats

As long as the time series database provides rest apis, the problem of being generic is resolved if we go with the approach that we took in USM 2.0 as below:

- ★ Every time series metric is named as per the convention
  - <parent_name{1..n}>.<entity_name>.usage_type
- ★ The user can then make the rest query as :
  - https://{server_ip}:{port}/monitoring?resource=<parent_name{1..n}>.<entity_name>.usage_type&from=<from_time>&until=<until_time> to get the statistics
  - In the above api, the optional parameters after /monitoring? Are used as it is by the api layer to make a http request to time series db, fetch response and directly return the same to the api user.

Notification Subscriptions @ Cluster level:

The central Monitoring application will expose apis to manage what types of alerts can be sent to the user and to whom(only users in tendrl will be allowed) and these subscription details will then be maintained internal to tendrl's central sore.

Status/Availability Monitoring:

1. **Process status monitoring:**
    a. Systemd on the nodes provides events when any of the systemd maintained processes are down which are notified to the sds-bridge via the socket exposed by it.
    b. The sds-bridge then makes appropriate state changes in the central store to reflect the same.
    c. Alerts can then be dispatched accordingly by the alerting module.
2. **Resource status monitoring:**
    a. Sds-bridge supports events(is the source in cephs case and a subscribes to events in gluster's case) for resource status changes.
    b. On detecting a state change through an event in sds-bridge, the sds-bridge makes corresponding state changes in the tendrl's central store.
    c. The state changes of certain collections are watched by the alerting module which sends notifications to the configured destinations

.

MONITORING PACKAGE INSTALLATION FLOWS

1. Tendrl core will maintain a configuration field(in central store) to indicate whether or not the monitoring is required. This field is false by default to indicate that monitoring is not required. This field is set by monitoring application(once installed and corresponding service started) using a rest api exposed by tendrl core for the same.
2. The physical resource related monitoring packages will be installed on all the nodes during the stage of initiating a node to be managed by tendrl.
3. The sds specific monitoring packages will be installed on the required nodes at cluster creation time if the monitoring_required field is true in the central store.

DISABLING MONITORING

1. Tendrl core provides a rest api to enable/disable the flag monitoring_required.
2. If the api is used to disable monitoring application, it does the following:
    a. Stop the monitoring application service.
    b. Stop collectd on all nodes managed by tendrl

Enabling Monitoring

1. Tendrl core provides a rest api to enable/disable the flag monitoring_required.
2. If this api is used for enabling the monitoring, skyring will then install monitoring application(if not already installed) and start the application(service).
3. It will also configure and start collectd's on all the nodes managed by it and in accordance with the principles of sds-bridge.(i.e, logical resources configured only on leader mon in case of ceph and a random node in case of gluster and these choices are persisted to the central store). This will be a clean up of a designated path of configurations(intended to be maintained only by tendrl's monitoring stack) and then followed by fresh generation of plugin configurations using the states in central store.

Note:

1. Stopping the monitoring application service external to tendrl service will only disable the alerting and aggregaion functionalities.
2. An ideal approach to completely opt out monitoring is to use the tendrl exposed rest api to disable monitoring.