Tigase Advanced Clustering Strategy (ACS)

Tigase Team

Tigase Advanced Clustering Strategy (ACS)

Tigase Team

Table of Contents

Chapter 1. Design and implementation

ACS

ACS is our general purpose, commercial clustering strategy designed for more or less typical XMPP installations easily scaling to millions and beyond of online users without limit on cluster nodes. The load tests we have run over the code were included a user database with 100mln accounts and an average roster size of up to 150, but that’s not the limit.

Design

The clustering strategy is based on sharing information between cluster nodes about online users. Who is online and where the user is connected. Communication between cluster nodes is processed with the highest priority to ensure minimal delays with online user data population. An efficient synchronization mechanism allows for a minimal traffic between cluster nodes and distributing accurate data about connecting and disconnecting users.

Chapter 2. Tigase ACS SM Installation

Tigase ACS SM component is by default provided with Tigase XMPP Server release (@-dist-max@ flavour of archive) so it's enough to enable it in the configuration. It can be also obtained from `tigase-acs` distribution package.

After downloading the archive it's simply matter of extracting it and copying contents of `jars/` directory of extracted archive to the `jars/` directory in `tigase-server/` installation directory, eg. under *nix systems (assuming the archive was downloaded to main Tigase Server directory):

```
tar --xf tigase-acs-${version}.tar.gz
cp --R tigase-acs-${version}/jars/ tigase-server/jars/
```

Chapter 3. Tigase ACS SM Configuration

In order to user Advanced Clustering Strategy, clustering mode first needs to be turned on:

```
'cluster-mode' = true
```

and then an ACS strategy needs to be enabled:

```
'sess-man' {
    strategy (class: tigase.server.cluster.strategy.OnlineUsersCachingStrategy) {}
}
```

Chapter 4. Supported components

Tigase Advanced Clustering Strategy for Multi User Chat (ACS-MUC)

Tigase Team <team@tigase.com [mailto:team@tigase.com]> v3.1.0, 2020-08-05 :numbered:

Overview

ACS for MUC allows seamless clustering of MUC rooms across Tigase XMPP server cluster installation. ACS for MUC is required for clustered MUC deployments. If offers various strategies to handle distribution of traffic and rooms across the cluster, which allows fine-tune configuration of the deployment to individual needs.

Tigase ACS MUC Configuration

In order to use ACS for MUC, main Advance Clustering Strategy is required. Once it’s enabled, clustered version of MUC component will be selected by default during startup therefore it’s not required to configure it explicitly (make sure no class is configured).

```
muc () {}
```

It’s also possible to explicitly configure the class with the following configuration:

```
muc (class: tigase.muc.cluster.MUCComponentClustered) {}
```

With the above configuration default MUC clustering strategy will be used. In order to select different strategy you have to configure it’s class in `strategy` bean within `muc` component bean:

```
muc () {
    strategy (class: tigase.muc.cluster.ShardingStrategy) {}
}
```

ACS MUC Strategies

ShardingStrategy

This is default clustering strategy used by Tigase ACS - MUC component. It should be used in most cases when we do not have small number of rooms with many occupants.

Short description

This is default clustering strategy used by Tigase ACS - MUC component. In this strategy MUC rooms are partitioned so each room is hosted only on one node. Every node contains full list of rooms with list of occupants available in each room and map which contains room address as a key and node as a value. If room is already opened (hosted) on some node, then node hosting this room is resolved using map of room to node. In other case node is selected by hash of room address using following algorithm

```
Math.abs(roomJid.hashCode()) % connectedNodes.size()
```

which gives a index of node on connected nodes list. So if node is not opened, then every cluster node should forward packets related to this room to the same node.

Connection to cluster

Once node connects to a cluster then map of hosted rooms and it’s occupants is synchronized between connecting node and any other already connected node.

Disconnection from cluster

If node is disconnected from a cluster due to server failure, then every other node will send "kick" stanzas to every occupant of every room hosted on disconnected node.

If node is disconnected due to node shutdown then node which is shutting down will send "kick" stanzas on it’s own, but it will notify every node about shutdown before disconnection from cluster.

Configuration

This is default strategy thus it’s used if no strategy configuration is present, but if you wish to enable this strategy and keep it enabled even if deafult clustering strategy will change in the future then you need to set `class` property of `strategy` bean within `muc` component to `tigase.muc.cluster.ShardingStrategy`.

Example:

```
muc () {
    strategy (class: tigase.muc.cluster.ShardingStrategy) {}
}
```

ClusteredRoomStrategy

This is clustering strategy which can be used for by Tigase ACS - MUC component, which is recommended for installations with relatively few rooms but rooms itself having a lot of occupants.

Short description

In this strategy MUC rooms are persistent and each room is hosted on every node. Every node contains full list of rooms (as they are persistent). Room on each node has knowledge only about occupants which joined room on this node. If remote user has joined room on one node and then it’s packets are delivered by S2S to other node, then packets will be forwarded to node to on which user joined room. Packets from user are processed by node on which they joined to room and notifications about joining room, leaving room, change of presences or about new message are sent to all other nodes and those other nodes are responsible for delivering proper notifications to users which joined room on them.

Connection to cluster

Once node connects to a cluster then map of occupants and their rooms are synchronized with other nodes.

Disconnection from cluster

If node is disconnected from a cluster, then every other node will send "kick" stanzas to every occupant of every room hosted on disconnected node.

Configuration

To enable this strategy you have to set `class` property of `strategy` bean within `muc` component to `tigase.muc.cluster.ClusteredRoomStrategy`.

Example:

```
muc () {
    strategy (class: tigase.muc.cluster.ClusteredRoomStrategy) {}
}
```

ClusteredRoomStrategyV2

This is clustering strategy which can be used for by Tigase ACS - MUC component, which is recommended for installations with relatively few rooms but rooms itself having a lot of occupants - contains improvements over ClusteredRoomStrategy

Short description

In this strategy MUC rooms are persistent and each room is hosted on every node. Every node contains full list of rooms (as they are persistent). Room on each node has knowledge only about occupants which joined room on this node. If remote user has joined room on one node and then it's packets are delivered by S2S to other node, then packets will be forwarded to node to on which user joined room. Packets from user are processed by node on which they joined to room and notifications about joining room, leaving room, change of presences or about new message are sent to all other nodes and those other nodes are responsible for delivering proper notifications to users which joined room on them.

This version contains improvements over the section called "ClusteredRoomStrategy" which leads to improved performance and reduced trafic over cluster connection due to changes in logic of processing packets.

Connection to cluster

Once node connects to a cluster then map of occupants and their rooms are synchronized with other nodes.

Disconnection from cluster

If node is disconnected from a cluster, then every other node will send "kick" stanzas to every occupant of every room hosted on disconnected node.

Configuration

To enable this strategy you have to set `class` property of `strategy` bean within `muc` component to `tigase.muc.cluster.ClusteredRoomStrategyV2`.

Example:

```
muc () {
    strategy (class: tigase.muc.cluster.ClusteredRoomStrategyV2) {}
}
```

Tigase Advanced Clustering Strategy for PubSub (ACS-PubSub)

Tigase Team <team@tigase.com [mailto:team@tigase.com]> v3.1.0, 2020-08-05 :numbered:

Overview

ACS for PubSub allows seamless clustering of PubSub nodes across Tigase XMPP server cluster installation. ACS for PubSub is required for clustered PubSub deployments. If offers various strategies to handle distribution of traffic and nodes across the cluster, which allows fine-tune configuration of the deployment to individual needs.

Tigase ACS PubSub Configuration

In order to use ACS for PubSub, main Advance Clustering Strategy (ACS) is required. Once it's enabled, clustered version of PubSub component will be selected by default during startup therefore it's not required to configure it explicitly (make sure no other class is configured).

```
pubsub () {}
```

It's also possible to explicitly configure the class with the following configuration:

```
pubsub (class: tigase.pubsub.cluster.PubSubComponentClustered) {}
```

With the above configuration default ACS PubSub clustering strategy will be used. In order to select different strategy you have to configure it's class for `strategy` bean within `pubsub` component bean:

```
pubsub () {
    strategy (class: tigase.pubsub.cluster.PartitionedStrategy) {}
}
```

ACS PubSub Strategies

ACS-PubSub Partitioned Strategy

Short description

This is default stategy used by Tigase ACS - PubSub component in which particular PubSub node is handled by only one cluster node.

Description of processing

In this strategy all configuration of nodes of the same PubSub service JID is done by the same node of a cluster which is dynamically selected based on hash of service JID and number of connected nodes within cluster. After any change to node configuration is done, then node established for processing manipulation of this particular PubSub node is notified about details of the PubSub node and detailed changes made to it's configuration. Remaining nodes do not require to know about changes.

Presences sent from users to PubSub service will be distributed to every node of a cluster.

Presence and IQ stanzas will be processed according to rules outlined below.

Rules of processing packets:

- `presence` - packets are delivered to every cluster node as each cluster node is responsible for handling different PubSub nodes
- `message` - is always processed locally
- `iq` - cluster node which will process this packet is selected based on following rules:
  - non-PubSub packets (no `pubsub` subelement) are processed on local node

- PubSub related packets without PubSub node name or packets that change PubSub node configuration (and contains `node` name and one of the following subelements: `create`, `configure`, `default`, `delete`) are processed on cluster node selected on hashcode derived from `to` attribute (service JID) and number of cluster nodes (this is done deal with concurrency issues between configuration changes)
- remaining PubSub pockets (non-configuration and containing `node` name) are processed on cluster node selected based on `to` attribute of packet and name of PubSub node

Note

This strategy for every packet (with exception of `presence`) should force processing of packet on only one cluster node (`presence` will be processed on all nodes)

Note

This strategy will react on PubSub configuration node change and will send notification to cluster node responsible for processing items for PubSub node (selected based on `to` attribute of packet, name of PubSub node) to trigger cached PubSub node configuration refresh

Note

Result of packets generated on remote node should **not be** filtered, so if packet from one cluster node was forwarded to other cluster node for processing, then response packet should not be filtered when this PubSub clustering strategy is used.

ACS-PubSub Clustered Node Strategy

Short description

This stategy can be used by Tigase ACS - PubSub component in which each PubSub node is handled on every cluster node but each cluster node will contain only partial information about user connections. This way strategy is better suited for deployments with PubSub nodes having a lot of subscribers. The benefit of using ClusterNodeStrategy in this case is reduced network traffic on cluster connections, as most of notifications and retrieval of items will be handled on the same cluster node to which user is connected.

Description of processing

In this strategy all configuration changes of pubsub nodes within same PubSub service JID are handled on the same node of the cluster (dynamically selected based on hash of service JID and number of connected nodes). After making changes to node configuration every cluster node is notified about this particular node identification (node name as well as service JID to which it belongs) and they refresh it’s cached configuration.

Presences sent from users to PubSub service will be delivered to every node of a cluster.

If every other IQ stanza sent to PubSub service which is does not change configuration of a node and is item publication stanza then this stanza will always be processed on local node (as data retrieval/removal may be done only on one node as items are not cached).

For IQ stanzas which are stanza responsible for publication of an item is forwarded for processing to every cluster node as only in this case PubSub will able to properly send notifications (as notification are generate always on local user node which reduces cluster network traffic).

Rules of processing packets:

- `presence` - packets are delivered to every cluster node as each cluster node is responsible for handling different PubSub nodes
- `message` - is always processed locally
- `iq` - cluster node which will process this packet is selected based on following rules:
  - non-PubSub packets (no `pubsub` subelement) are processed on local node
  - PubSub related packets without PubSub node name or packets that change PubSub node configuration (and contains `node` name and one of the following subelements: `create`, `configure`, `default`, `delete`) are processed on cluster node selected on hashcode derived from `to` attribute (service JID) and number of cluster nodes (this is done deal with concurrency issues between configuration changes)
  - PubSub item publication packets (must contain `node` name attribute within `pubsub` element) are delivered to all cluster nodes and each cluster node generates notifications about item publication to users connected to that particular cluster node.

Note

This strategy will react to PubSub configuration node change and will send notification to every cluster node (as every cluster node is responsible for processing items for every PubSub node) to refresh particular PubSub node configuration.

Note

Result of processing of packets generated on remote node should **be** filtered if packet was also processed on local node, so if packet from one cluster node was forwarded to other cluster node for processing but was not processed locally, then response packet should be filtered when this PubSub clustering strategy is used.

In a nutshell

`Partitioned Strategy` assigns PubSub node to cluster node and handle all processing on that particular cluster node (configuration change, generating notification packets), `Clustered Node Strategy` distributes processing of nodes across cluster - each cluster node has complete knowledge of all PubSub nodes but performs processing that corresponds only to itself (i.e. generates notifications only for users connected to this particular node).

Tigase Advanced Clustering Strategy for WorkGroup (ACS-WG)

Tigase Team <team@tigase.com [mailto:team@tigase.com]> v3.1.0, 2020-08-05

Overview

ACS for WorkGroup allows seamless clustering of WorkGropu nodes across Tigase XMPP server cluster installation. ACS for WorkGroup is required for clustered WorkGroup deployments. If offers various strategies to handle distribution of traffic and nodes across the cluster, which allows fine-tune configuration of the deployment to individual needs.

Tigase ACS WorkGroup Configuration

In order to use ACS for WorkGroup, main Advance Clustering Strategy (ACS) is required. Once it's enabled, clustered version of WorkGroup component will be selected by default during startup therefore it's not required to configure it explicitly (make sure no other class is configured).

```
wg () {}
```

It's also possible to explicitly configure the class with the following configuration:

```
wg (class: tigase.workgroupqueues.cluster.WorkgroupQueuesClusteredComponent) {}
```

With the above configuration default ACS WorkGroup clustering strategy will be used. In order to select different strategy you have to configure it's class in `strategy` bean within `wg` component bean:

```
wg () {
    strategy (class: tigase.workgroupqueues.cluster.ClusteredStrategy) {}
}
```