MSGorilla Infrastructure

MSGorilla is a twitter-like engineering event system. The infrastructure of MSGorilla is mainly built on azure storage and thus should be easily moved to woss storage.

The components used in MSGorilla including:



* + Monitors: Collect information from other system and report to MSGorilla through MSGorilla Rest API (MSGorilla SDK is also available to call rest API).
  + Users: Choose monitors to follow and read messages from the MSGorilla Website.
  + Woss Table: All messages are stored in Woss(Azure) table
  + Sql Server: User and group info are managed are stored in sql server for there’re some join actions and are not suitable for table service.
  + Message Dispatcher: Processing messages post by monitor. Generally insert messages into topicline, homeline and etc.
  + Indexer: For message searching.

A sample message contains the following key info:

{

"User": "WossWAESMonitor",

"ID": "251985830149946\_woss\_WossWAESMonitor\_51dd5cfb-cb13-4f06-b558-1a8b8f337903",

"Group": "woss",

"SchemaID": "none",

"Owner": [

"eridai"

],

"AtUser": null,

"TopicName": [

"WAES Job 1-d7de33ec-8ce1-43c4-bddd-299a93e71926-eridai",

"WOSS WAES Job"

],

"MessageContent": "WAES job status changed to InProgress from Starting \n #WAES Job 1-d7de33ec-8ce1-43c4-bddd-299a93e71926-eridai#",

"PostTime": "2014-11-20T08:04:10.0530449Z",

"Importance": 2

}

* User: The user ID who post the message
* ID: Message ID which is composed of timestamp, group id, user id and a guid
* Group: Group ID. Messages in private group should not be viewed by users who are not in the group
* SchemaID: Specify the message type, text or diagram(the web front end will try to draw diagram on the web page for such messages).
* Owner: The user id of the message owner
* AtUser: The users’ id of related user
* TopicName: The topic names of the related topic
* MessageContent: The text content of the message
* PostTime: The timestamp frontend received the message.
* Importance: A message with importance 0 will be tagged Important on the websit frontend.

When a new message reached the MSGorilla front end, it will be processed in the following ways:



A message will be duplicated and stored in tables including Userline, HomeLine, Topicline, Ownerline, Atline and Publicline so that frontend can quickly query different table for different requirement. The schema of these table are the same, only the partition keys and row keys are set in different ways.

|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Partition Key | Row Key | Description |
| UserLine | {Post User ID}\_{Date} | {Group ID}\_{Msg ID} | The partition key contains the user who post message. Thus user can find what they have posted here. |
| HomeLine | {Follower ID}\_{Date} | {Group ID}\_{Msg ID} | The user id in partition key are the followers. If one post a message and he has N followers, then N copy of the message will be stored in HomeLine with different follower ID. Thus user can get messages in homepage by querying this table |
| TopicLine | {TopicID}\_{Date} | {Msg ID} | Querying this table to get messages in the same topic |
| OwnerLine | {Owner User ID}\_{Date} | {Msg ID} | The user id is the owner of the message |
| AtLine | {At User ID}\_{Date} | {Msg ID} | The user id are related users of the message |
| PublicLine | {Group ID}\_{Date} | {Msg ID} | Query this table to get latest messages in a certain group |

Message ID is started with the timestamp so that messages in the same partition are order by timestamp.

When a new message reached the MSGorilla front end, front end will directly insert the message into UserLine and UnprocessedMessage table. UnprocessedMessage table works as a queue (actually in azure implementation it’s a queue) and dispatcher get messages from UnprocessedMessage table and further insert the copy into other tables.

Indexer is running for text searching service. The UnindexedMessage table works as UnprocessedMessage and indexer get new messages from it and then processing and insert data into wordindex table.