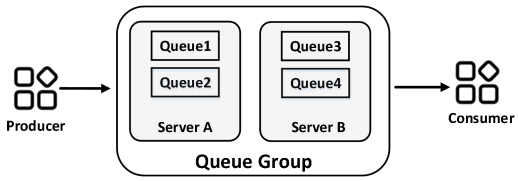

Fig. 1. Message queue group model

A. *Cluster strategy*

Most messaging middleware products support clustering of message servers to ensure the reliability of the message source. Each messaging server in the cluster has a separate message queue with detailed message distribution capabilities. The existing clustering scheme is divided into high performance and high-availability. The superior performance scheme aims to provide higher throughput in unified service time. The high-availability scheme aims to reduce the outage time of the entire message service. In the SIS integrated application environment, the primary goal of message service design is to make sure the normal operation of the service and provides highly reliable disaster recovery performance. Existing message middleware products [15-17] provide a variety of high-availability cluster strategies, which are mainly divided into two parts: service processing node backup and data backup. Service processing node backup is used to back up the message forwarding function in the message server. Data backup is the backup of messages in the message queue. Different clustering strategies can be used in different application scenarios.

In SIS, service processing node backups and data backups are equally important. However, the backup strategy is closely related to the size of the hardware in the cluster. Considering the limited performance of hardware devices in offshore environments, it is unacceptable to use full backups for service processing nodes and data. In order to guarantee the disaster tolerance of the service node, we hope that the service processing node uses a full backup strategy. On the other hand, message queue data use an incomplete backup strategy to reduce the redundancy of data.

Combined with the message queue group model, we propose a new cluster strategy, as shown in Figure 4. Multiple message queue entities in a message queue group have the same message data, enabling redundant backup of data. At the same time, in order to reduce the redundancy of data, the number of message queue entities in the message queue group Nq ⩽ the number of message servers Ns, and distributed in different message servers. When Ns=1or 2, Nq=Ns. When Ns ⩾3, Nq⩽ Ns, Nq is adjusted according to the message load condition in the message queue group.

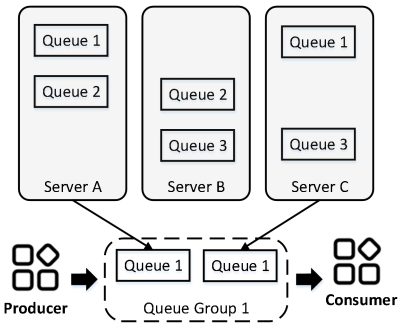

Fig. 2. Cluster strategy diagram

B. *Message Server Monitoring Module*

In order to monitor the real-time status of the message server and message queue, we designed the message server monitoring module. The monitoring module runs independently in each message server to obtain various information of each message server node, including system status, AMQ Broker status, and message queue status. The monitoring module is mainly divided into Java Management Extensions (JMX) management module, Sigar module, and Dubbo module. As is shown in Figure 5.

Among them, the JMX management module uses the JMX extension interface provided by AMQ to obtain and control the running status information of the Broker and the message queue. Sigar is the primary data collection component of Hyperic HQ and can be used to collect information from multiple operating systems [18]. We use the Sigar tool to get system status information such as CUP, disk, network I/O, memory, and JVM. Dubbo is a Java-based Remote Procedure Call (RPC) framework that provides interface-based remote calls and automatic service registration and discovery [19].

The monitoring module uses Dubbo to provide remote monitoring and control interfaces for the configuration center. The configuration center uses these interfaces to manage the message server and message queue uniformly. On the other hand, the function of automatic service registration and discovery by using Dubbo is used to achieve flexible expansion of the message server.