IceStorm消息运行过程：消息发布者(publisher)将消息发送到IceStorm, IceStorm再将该消息发送给所有订阅该主题的接收者(subscriber)

一个消息发布者可以发布多个主题的消息,一个消息订阅者也可以订阅多种类型的消息，就像微博用户可以关注多个主题一样，当消息发送者投递消息到storm服务器时，storm会转发该消息

执行过程：

1. 启动icebox服务,所需配置文件为 config.icebox 和 config.service

config.icebox中指定storm服务的地址

IceBox.ServiceManager.Endpoints=tcp -h xiangzhenwei.peraportal.com -p 9998

config.service中指定storm服务示例的名字

IceStorm.InstanceName=DemoIceStorm

config.service中指定storm监听的地址

IceStorm.TopicManager.Endpoints=default -h xiangzhenwei.peraportal.com -p 10000

启动服务需要一个缓存目录db，缓存主题相关信息，该目录名称在config.service文件中定义，如下：Freeze.DbEnv.IceStorm.DbHome=db,启动服务前保证配置文件所在目录存在该文件夹

启动icebox：打开一个命令窗口，切换到 config.icebox 和 config.service

所在目录，输入 icebox --Ice.Config=config.icebox 回车

1. 启动消息发布者和订阅者

消息订阅者和发布者都需要连接到storm服务器，因此需要在代码或配置文件中配置storm的地址：

下面是一个简单的接口News，演示新闻消息的分发过程。

StormNewsDemo.ice文件如下：

#pragma once

module Pera

{

interface News

{

void NewMsg(string time);

};

};

新闻消息发布者代码如下：

#include <IceUtil/IceUtil.h>

#include <Ice/Ice.h>

#include <IceStorm/IceStorm.h>

#include <StormNewsDemo.h>

using namespace std;

using namespace Pera;

void main()

{

Ice::InitializationData initData;

initData.properties = Ice::createProperties();

initData.properties->setProperty("Ice.MessageSizeMax", "102400" );//默认是1024，单位KB

initData.properties->setProperty("Ice.ThreadPool.Server.Size", "1");

initData.properties->setProperty("Ice.ThreadPool.Server.SizeMax", "1000" );

initData.properties->setProperty("Ice.ThreadPool.Server.SizeWarn", "1024");

Ice::CommunicatorPtr communicatorPtr = Ice::initialize(initData);

// icestorm的地址"StormNewsDemo/TopicManager:tcp -h xiangzhenwei.peraportal.com -p 10000"

IceStorm::TopicManagerPrx manager = IceStorm::TopicManagerPrx::checkedCast(

communicatorPtr->stringToProxy("StormNewsDemo/TopicManager:tcp -h 192.168.23.129 -p 10000"));

IceStorm::TopicPrx topic;

try

{

topic = manager->retrieve("news");

}

catch(const IceStorm::NoSuchTopic&)

{

try

{

topic = manager->create("news");

}

catch(const IceStorm::TopicExists&)

{

cerr << "NewsPub.exe" << ": temporary failure. try again." << endl;

return;

}

}

//

// Get the topic's publisher object, and create a Clock proxy with

// the mode specified as an argument of this application.

//

Ice::ObjectPrx publisher = topic->getPublisher();

publisher = publisher->ice\_oneway();

NewsPrx news = NewsPrx::uncheckedCast(publisher);

cout << "publishing tick events. Press ^C to terminate the application." << endl;

try

{

while(true)

{

news->NewMsg("现在时间：" + IceUtil::Time::now().toDateTime());

IceUtil::ThreadControl::sleep(IceUtil::Time::seconds(1));

}

}

catch(const Ice::CommunicatorDestroyedException&)

{

// Ignore

}

}

新闻消息订阅者代码如下：

#include <IceUtil/IceUtil.h>

#include <Ice/Ice.h>

#include <IceStorm/IceStorm.h>

#include <StormNewsDemo.h>

using namespace std;

using namespace Pera;

class NewsI : public News

{

public:

// 新闻消息订阅者重写该函数，展示收到的消息。

virtual void

NewMsg(const string& msg, const Ice::Current&)

{

cout << msg << endl;

}

};

void main()

{

Ice::InitializationData initData;

initData.properties = Ice::createProperties();

initData.properties->setProperty("Ice.MessageSizeMax", "102400" );//默认是1024，单位KB

initData.properties->setProperty("Ice.ThreadPool.Server.Size", "1");

initData.properties->setProperty("Ice.ThreadPool.Server.SizeMax", "1000" );

initData.properties->setProperty("Ice.ThreadPool.Server.SizeWarn", "1024");

Ice::CommunicatorPtr communicatorPtr = Ice::initialize(initData);

// icestorm的地址"StormNewsDemo/TopicManager:tcp -h xiangzhenwei.peraportal.com -p 10000"

IceStorm::TopicManagerPrx manager = IceStorm::TopicManagerPrx::checkedCast(

communicatorPtr->stringToProxy("StormNewsDemo/TopicManager:tcp -h 192.168.23.129 -p 10000"));

if(!manager)

{

cerr << "NewsSub.exe" << ": invalid proxy" << endl;

return;

}

IceStorm::TopicPrx topic;

try

{

topic = manager->retrieve("news");

}

catch(const IceStorm::NoSuchTopic&)

{

try

{

topic = manager->create("news");

}

catch(const IceStorm::TopicExists&)

{

cerr << "NewsSub.exe" << ": temporary failure. try again." << endl;

return;

}

}

// 接收端监听消息的地址"tcp -h 0.0.0.0:udp -h 0.0.0.0"

string strEndPoint = "tcp -h 0.0.0.0:udp -h 0.0.0.0";

Ice::ObjectAdapterPtr adapter = communicatorPtr->createObjectAdapterWithEndpoints("News.Subscriber", strEndPoint );

//

// Add a servant for the Ice object. If --id is used the identity

// comes from the command line, otherwise a UUID is used.

//

// id is not directly altered since it is used below to detect

// whether subscribeAndGetPublisher can raise AlreadySubscribed.

//

Ice::Identity subId;

subId.name = IceUtil::generateUUID();

Ice::ObjectPrx subscriber = adapter->add(new NewsI, subId);

//

// Activate the object adapter before subscribing.

//

adapter->activate();

subscriber = subscriber->ice\_oneway();

IceStorm::QoS qos;

qos["retryCount"] = 3;

try

{

topic->subscribeAndGetPublisher(qos, subscriber);

}

catch(const IceStorm::AlreadySubscribed&)

{

// If we're manually setting the subscriber id ignore.

cout << "reactivating persistent subscriber" << endl;

}

communicatorPtr->waitForShutdown();

topic->unsubscribe(subscriber);

}