# 需求描述

概述:在对角街和云点项目中，涉及到定时任务调度，主要包括数据库更新、缓存更新、email|短信通知，消息推送。下面将需要定时任务的需求进行详细描述。

## 对角街

* 商品发布,指定时间上架
* 商品发布,超过有效期自动下架
* 众销购订单状态为:等待买家付款,如果超过7日未付款,则交易自动关闭
* 众销购订单状态为:等待卖家发货,如果超过7日未发货,则交易自动关闭,同时将货款、快递费退到买家的资金账户
* 众销购订单状态为:卖家已发货(等待买家确认收货),如果超过7日买家未确认收货,则自动进入下一个订单状态:交易成功，同时将货款转到卖家的资金账户
* 不退货退款流程，当买家提交退款申请，状态变为:等待买家同意退款，如果3日卖家未点击同意按钮，则自动退款.
* 需要退货退款流程，当买家提交退货退款申请，状态变为: 等待卖家同意退货退款,如国5日卖家未点击同意按钮，则自动变为下一个状态:同意退货退款
* 需要退货退款流程，等待卖家收货并退款，如果卖家在7日内未收货，则自动将钱给买家，同时状态更新为退货成功退款成功，如果卖家点击未收到退货，则延迟3天。

## 云点

待补充

# 逻辑架构



## web应用服务器

指对角街或云点项目背后真实的web应用服务器。

* Web应用服务器负责生成待执行的原始任务消息，放入到activeMQ-server上。

## activeMQ-server(M/S)

jms服务器，采用队列模式。为了解决单点故障采用master/slave的工作模式。

* 接收web应用服务器的原始消息，并将原始消息推送给调度器。
* 接收调度器的待调度的任务消息,并该将消息推送给指定的任务执行服务器。
* 接收任务执行服务器的调度中和执行结束的消息,并该将消息推送给调度器。

## 调度器

* 提供mq消息监听，负责接收activeMQ-server发送过来的原始任务消息，按照“分配算法”指定消息的任务执行服务器，并将消息发送到activeMQ-server，同时插入数据库。接收activeMQ-server发送过来的调度中的任务和执行完成后的任务，更新数据库状态。对于依赖的任务也可以触发。(对于失败的任务需要告警)
* 提供注册服务，任务执行服务器启动时调用调度器的注册服务，将该任务执行服务器的服务信息加入到服务列表。(也可以通过点击button，手工调用注册服务)
* 提供注销服务，在任务执行服务器停机或重启的时候调用调度器的注销服务，将该任务执行服务器从服务列表删除。调度服务器读取mqserver上当前服务器对应的待调度的任务、调度中的任务，重新进行任务分配。
* 提供心跳服务，判断任务执行服务器是否正常。如果不正常，调度服务器读取mqserver上当前服务器对应的待调度的任务、调度中的任务，重新进行任务分配。

## 任务执行服务器

* 负责接收mq传送过来的消息
* 解析消息
* 放入quartz
* 任务执行完成后，将任务执行的结果发送mq-server。

# 详细设计

## 消息字段

taskId:当前的任务id

dependenceTaskId:依赖的任务id

taskType:任务类型(updateDB、updateCache、messageNotify、messagePush)

taskAction:任务操作类型(new、cancel、update)

new:新增的1个单独执行的任务，不依赖其他任务.

如果为空，就是1个普通的任务, 当dependenceTaskId不为空时，表示需要当

dependenceTaskId对应的任务执行完成后，才执行当前任务

cancel:取消某个任务, dependenceTaskId表示需要取消的任务id

update:更新某个任务,如更新调度时间，dependenceTaskId表示需要更新的任务id

serverName任务执行服务器(ip:port)。

className,包名+类名

method:通过反射机制待调用的方法

args:调用method时，传入的参数，有可能多个参数，暂用数组的方式”[{“java.lang.Integer”:1},{“java.lang.String”:”a”}]”}”

具体开发的时候，可以根据具体情况做调整.

通过反射动态调用代码参考:

http://hi.baidu.com/kekeyesterday/item/4860e7d51caae6be32db9066

executeTime:执行时间

interval\_time:间隔时间(秒)

executeCounts:执行次数

taskStatus:任务状态(1:待调度 2.调度中 3:执行成功 4:执行失败).

waitSchedule、scheduling、success、fail

建议:taskId, serverName, taskType, executeTime, taskStatus考虑放在property中

更新任务的字段信息，没有考虑周全，如更新时间

## 应用服务器发送MQ消息

Web应用服务器生成任务消息发送到activemq-server

***参考代码***

ConnectionFactory cf = new ActiveMQConnectionFactory("tcp://ip:61616");

Connection conn = null;

Session session = null;

try {

conn = cf.createConnection();

session = conn.createSession(false, Session.AUTO\_ACKNOWLEDGE);

Destination destination = new ActiveMQQueue("diagonally-queue");

MessageProducer producer = session.createProducer(destination);

TextMessage message = session.createTextMessage();

String messageInfo = "{“taskId”:null,”taskType”:1,…….}";//上面定义好的消息格式

message.setText(messageInfo);

producer.send(message);

} catch (JMSException ex)

{

ex.printStackTrace();

} finally {

try {

if(session != null) {

session.close();

}

if(conn != null) {

conn.close();

}

} catch (JMSException ex)

{

ex.printStackTrace();

}

}

## 调度器接收MQ消息

***参考代码***

ConnectionFactory factory = new ActiveMQConnectionFactory("tcp://ip:61616");

Connection connection = factory.createConnection();

connection.start();

Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

Destination destination = new ActiveMQQueue("diagonally-queue");

MessageConsumer consumer = session.createConsumer(destination);

consumer.setMessageListener(new MessageListener() {

public void onMessage(Message message) {

TextMessage tm = (TextMessage) message;

try {

**//根据“分配算法”指定serverName，发送到指定serverName的队列。**

**ScheduleTaskUtil.assignTaskMessageToMQServer(tm.getText());**

} catch (JMSException e) {

e.printStackTrace();

}

}

});

## 调度器与任务执行服务器的通信

### 注册服务

当任务服务器启动时，通过http协议向调度器注册当前任务服务器，则调度服务器在维护的server列表里面，将该server加进去。同时扫描由于没有任务服务器启动的原因对应的任务列表，发送给mq-server

备注:手工触发也可以，适用于调度服务器启动晚于真实的服务器,需要调度服务器提供页面button

### 心跳通知

任务执行服务器，每隔5秒钟时间通过http协议请求调度器的心跳通知url。

如果调度器接收到任务执行服务器的心跳请求，执行刷新调度器的服务器列表操作(如果有则不更新，没有则把该server加入服务列表)。

如果调度器超过15秒没有接收到任务执行服务器的心跳请求，则将该服务器从调度器的服务列表删除。调度服务器读取mqserver上当前服务器对应的待调度的任务、调度中的任务，重新进行任务分配。同时发邮件|短信通知管理员。

### 注销服务

当真实任务服务器出现重启、停机的情况下，当前server需要通过http协议主动通知调度器将当前server从列表里去掉，调度服务器读取mqserver上当前服务器对应的待调度的任务、调度中的任务，重新进行任务分配。

### 分配服务

概述:选择后台的哪个服务器进行任务分配时，首先调度器需要根据一定的“分配算法”进行任务调度。

**参考代码**

**//根据“分配算法”指定serverName，发送到指定serverName的队列。**

**public void assignTaskMessageToMQServer(String message){**

**//1).如果维护服务列表里没有任务server，则把当前message插入到数据库,并短信或邮件告知管理员**

**return;**

**//2).编写调度算法，确定需要发送任务的server**

**//3).将消息中的serverName字段进行更新**

**//4).将更新后的消息放入该server对应的队列。**

**}**

## 任务服务器获取任务并执行

任务服务器开启mq监听获取属于自己的任务

解析消息

待调度的任务加入到quartz调度框架中，同时将调度中的任务放入队列。

将执行结束后的任务放入队列。

数据库设计

sys\_schedule\_task

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **字段名称** | **中文说明** | **键别** | **是否空** | **数据类型** | **长度** | **备注** |
| task\_id | 任务id | PK | NO | varchar | 64 |  |
| dependence\_task\_id: | 依赖的任务 |  | YES | varchar | 64 |  |
| task\_type | 任务类型 |  | NO | VARCHAR | 16 | updateDB、updateCache、messageNotify、messagePush |
| task\_action | 任务动作 |  | YES | VARCHAR | 16 | new、cancel、update |
| server\_name | 任务执行服务器 |  | YES | VARCHAR | 32 | 当没有服务列表时，插入 |
| springContextBean | 执行的class |  | YES | VARCHAR | 128 | 如果为取消任务、更新任务，为空 |
| className | className |  | NO | VARCHAR | 256 | 待执行的对象对应的className |
| method\_name | 执行的方法 |  | YES | VARCHAR | 64 | 如果为取消任务、更新任务，为空 |
| args | 执行的参数 |  | YES | VARCHAR | 1024 | 如果为取消任务、更新任务，为空 |
| execute\_time | 开始执行时间 |  | YES | DATETIME | 256 | 第一次执行时间 |
| interval\_time | 间隔时间 |  | YES | INTEGER |  | 间隔时间(s) |
| execute\_counts | 执行次数 |  | YES | INTEGER |  | 执行次数 |
| task\_status | 执行状态 |  | NO | VARCHAR | 16 | waitSchedule、scheduling、success、fai |
| fail\_reason | 失败原因 |  | YES | VARCHAR | 1024 | 失败原因 |
| create\_time | 创建时间 |  | NO | DATETIME |  |  |
| update\_time | 修改时间 |  | YES | DATETIME |  |  |
| is\_valid | 是否有效 |  | NO | INTEGER |  |  |

# 进展

|  |  |  |  |
| --- | --- | --- | --- |
| 任务模块 | 任务 | 是否完成 | 说明 |
| 服务器搭建 | Web应用服务器 | 是 | 跟自己项目有关  如对角街、云点 |
| activeMQ服务器 | 是 | 192.168.1.181:61616  暂不支持(m/s) |
| scheduleServer(1台) | 是 | 总调度器 |
| taskServer(若干) | 是 | 真实的任务服务器 |
| 调度器 | 注册服务 | 是 | 1.任务服务器启动时注册  2.手工注册(适用于调度器启动早于任务服务器未做) |
| 注销服务 | 否 |  |
| 心跳服务 | 否 |  |
| 任务重新分配服务 | 否 | 当某个任务执行服务器出现异常或注销时，重新分配任务 |
| 任务依赖 | 否 |  |
| 任务异常处理 | 否 |  |
| 任务服务器 | 简单任务 | 是 | 支持在指定时间点执行，每隔多少秒执行，共执行多少次 |
| 日历任务 | 是 | 如在每个星期几几点执行 |
| 监控模块 | 任务服务器列表查看 | 否 | 包括任务服务器ip、服务器正在执行的任务、历史任务查看等 |
| 任务服务器页面注册 | 否 | 通过页面button，手工注册任务  (适用于调度器启动早于任务服务器) |
| 日历任务页面添加 | 否 | 适用于后台调度任务，不用各web应用服务器启动初始化的时候发消息给调度器 |
| 日历任务页面批量启动 | 否 | 适用于当所有的任务服务器停机的情况下，通过点击批量启动按钮，批量启动所有日历任务 |

# 实施

## 下载工程

从snv下载或oa传输

## 部署

### mqServer、队列配置

修改scheduleServer的mqserver下的com.atman.base.mqserver.Constant.java

包含:mq服务器的信息、队列名称

修改taskServer的mqserver下的com.atman.base.mqserver.Constant.java

包含:mq服务器的信息、队列名称、scheduleServer信息

### 新建schedule\_task表

现在scheduleServer的库里面建张任务表

CREATE TABLE `schudule\_task` (

`task\_id` VARCHAR(64) NOT NULL,

`dependence\_task\_id` VARCHAR(64) NULL DEFAULT NULL COMMENT '依赖任务id',

`task\_type` VARCHAR(16) NOT NULL COMMENT '任务类型',

`task\_action` VARCHAR(16) NOT NULL COMMENT '任务动作',

`server\_name` VARCHAR(16) NULL DEFAULT NULL COMMENT '分配的任务执行服务器',

`spring\_context\_bean` VARCHAR(45) NOT NULL COMMENT '分配的上下文bean',

`class\_name` VARCHAR(256) NOT NULL COMMENT 'className',

`method\_name` VARCHAR(45) NOT NULL COMMENT '待执行的方法',

`args` VARCHAR(1024) NULL DEFAULT NULL COMMENT '参数',

`execute\_time` DATETIME NULL DEFAULT NULL COMMENT '第一次执行时间',

`cron\_expression` VARCHAR(64) NULL DEFAULT NULL COMMENT '执行表达式',

`interval\_time` INT(11) NULL DEFAULT NULL COMMENT '间隔时间(s)',

`execute\_counts` INT(11) NULL DEFAULT NULL COMMENT '执行次数',

`task\_status` VARCHAR(16) NOT NULL COMMENT '任务类型',

`fail\_reason` VARCHAR(1024) NULL DEFAULT NULL COMMENT '失败原因',

`create\_time` DATETIME NOT NULL COMMENT '创建时间',

`update\_time` DATETIME NULL DEFAULT NULL COMMENT '修改时间',

`is\_valid` INT(2) NOT NULL COMMENT '是否有效',

PRIMARY KEY (`task\_id`)

)

COLLATE='utf8\_general\_ci'

ENGINE=InnoDB;

考虑到代码质量和项目之间的影响，先云点和对角机分开部署。

需要调度的任务对应的java类，打包放到taskServer的lib即可

### 新建测试表

**CREATE** **TABLE** `hello` (

`id` **BIGINT**(20) **NOT** **NULL** **AUTO\_INCREMENT**,

`name` **VARCHAR**(255) **NOT** **NULL**,

**PRIMARY** **KEY** (`id`)

)

**COLLATE**='utf8\_general\_ci'

**ENGINE**=**InnoDB**

**AUTO\_INCREMENT**=13;

## 注意事项

先启动scheduleServer工程、在启动taskServer工程.

以后如果有页面可以手工注册服务，就不存在启动顺序的问题.

## 调用举例

请参考scheduleServer工程下的ActivemqUtil.java的main()方法

由于调度server不成熟，提供给各个应用的调用jar包延迟给出