Rendszerintegráció és felügyelet laboratórium (VIMIM309)

Megbízható üzenetküldés Rabbit MQ alapon

Mérési segédlet

Készítette: Bozóki Szilárd

Utolsó módosítás: 2015. január 13.

Verzió: 1.0

Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

Bevezető és rövid ismertető

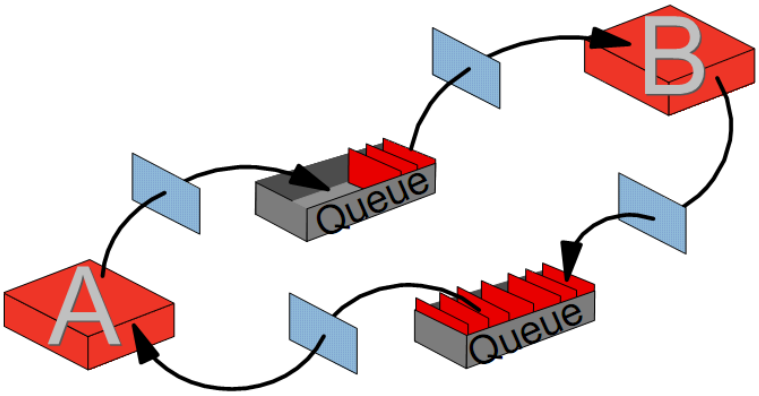
A mérés célja az üzenetsorokhoz kapcsolódó alapvető ismeretek elsajátítása. A mérés során a hallgatók a korábban elkészített Java alapú munkafolyamatot alakítják át üzenet alapú vezérlésűvé RabbitMQ használatával, ami egy Advanced Message Queuing Protocol (AMQP) implementáció.

Üzenetsorok

Egy üzenet alapú kommunikációs alrendszer célja, hogy kézbesítse a termelő (feladó, publisher) üzenetét (message) a fogyasztónak (consumer). Alap esetben az üzenetek a termelőktől direkt a fogyasztókhoz kerülnének, de üzenetsorok (queue) használata esetén az üzenetek egy soron keresztül indirekt kerülnek a fogyasztókhoz. Az üzenetsorok közbeiktatása révén így a termelő és a fogyasztó nincsenek direkt kapcsolatban és az üzenetek kézbesítés két lépésre bontható. Az egyik lépésben eljut az üzenet az üzenetsorig, a másik lépésben megérkezik a az üzenetsorból a fogyasztóhoz. Gyakorlatilag üzenetsorok használata során a termelő alkalmazás nem a konkrét fogyasztó alkalmazást célozza (címzi) meg, hanem egy cél üzenetsort.

Egy alkalmazásnak több bemeneti és több kimeneti sora is lehet, így a termelő fogyasztó szerepek természetesen keveredhetnek és sorról sorra változhatnak.

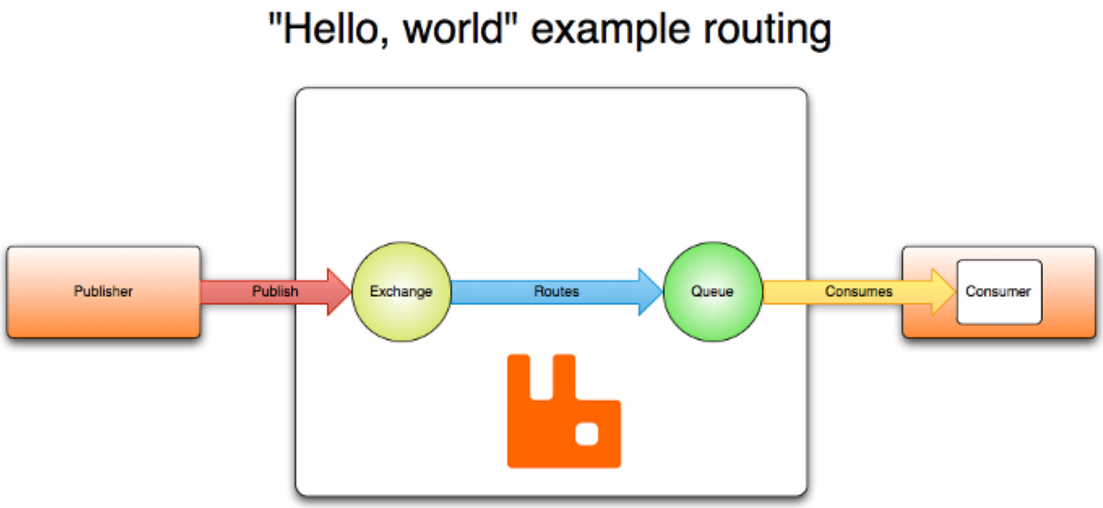
Üzenetsorok használata esetén a termelőnek azzal sem kell törődnie, hogy a cél fogyasztó elfoglalt vagy egyáltalán elérhető-e állapotban van e, ugyanis az üzenetküldő keretrendszer elintézi a célállomáshoz való szállítást.



1. ábra Üzenetek és sorok

Az ábrán két, egymással kommunikáló program látható, ahol az egyik kimeneti sora a másik bemeneti sora és fordítva, így a programok mind termelőként és fogyasztóként is viselkednek. A téglalapok a sorok és programok között a kommunikációs alrendszer interfészeit (API) reprezentálják, amelyeken keresztül a kommunikációs alrendszer és a programok érintkeznek.

RabbitMQ



2. ábra Az üzenetküldés AMQP modellje

AMQP esetén az üzeneteket az Exchange irányítja (route) az üzenetsorok felé. Egy üzenet több sorba is továbbítható, így az Exchange működése lehet például broadcast, unicast vagy multicast is.

Egy üzenetsorhoz több fogyasztó is tartozhat, ebben az esetben az üzenetsor beállítása határozza meg a kézbesítési logikát, ami lehet például round robin, ami az üzeneteket egyenletesen osztja el a fogyasztók között.

RabbitMQ eszközök

A rabbitmqctl a RabbitMQ legalapvetőbb eszköze. Olyan alapvető parancsokat lehet vele kiadni, mint például indítás, újraindítás, leállítás és a további pluginek engedélyezése.

A Management Plugin a RabbitMQ server kezelésére és monitorozására szolgál. Három felülettel is rendelkezik, amiket az adott szerveren különböző címeken lehet elérni.

1. **Grafikus webes** (pl. http://*server-name*:15672/)
2. **HTTP API** (pl. http://*server-name*:15672/api/)
3. **CLI** Management Command Line Tool : (pl. http://*server-name*:15672/cli/)

Hello World mintapélda

Termelő

A Hello World termelő oldali mintapéldája során először kapcsolódunk a RabbitMQ szerverhez és létrehozunk egy kommunikációs csatornát. Ezek után a csatornán deklarálunk egy üzenetsort egy választott névvel. Utána publikálunk egy rövid üzenetet és végezetül lezárjuk a csatornát és a kapcsolatot.

Kapcsolódás a RabbitMQ –hoz és csatorna felépítés

```
    ConnectionFactory factory = new ConnectionFactory();
    factory.setHost("localhost");
    Connection connection = factory.newConnection();
    Channel channel = connection.createChannel();
```

Kapcsolódás során először létrehozzuk a kapcsolat felépítő objektumot (ConnectionFactory). Ezek után beállítjuk rajta a RabbitMQ szerver címét, ami a jelen példa esetén „localhost”. Majd létrehozzuk a kapcsolatot (newConnection). Végezetül a kapcsolaton felépítünk egy csatornát.

Üzenetsor létrehozás

```
    channel.queueDeclare(QUEUE_NAME, false, false, false, null);
```

Üzenetsort a csatorna queueDeclare metódusával tudunk létrehozni. A metódus első paramétere a létrehozandó üzenetsor neve, ami majd azonosítani fogja az üzenetsort. Az üzenetsor neve jelen esetben a QUEUE_NAME változó értéke lesz.

Üzenet publikálás

```
    String message = "Hello World!";
    channel.basicPublish("", QUEUE_NAME, null, message.getBytes());
```

Üzenetküldéshez a basicPublish metódust használhatjuk. Címzéshez meg kell adni a csatorna nevét, ami jelen esetben a QUEUE_NAME változó értéke. Az üzenet tartalmát bájt tömbként (byte[]) lehet átadni, így a szerializálásról gondoskodnunk kell. (getBytes())

Csatorna és kapcsolat lezárása

```
    channel.close();
    connection.close();
```

Fogyasztó

A Hello World fogyasztó oldali mintapéldája során a termelőhöz hasonlóan először kapcsolódunk a RabbitMQ szerverhez és létrehozunk egy kommunikációs csatornát, majd után a csatornán deklarálunk egy üzenetsort a termelővel azonos névvel. Ezek után a csatornához hozzárendelünk egy puffer objektumot, amibe a RabbitMQ aszinkron (callback) kézbesíti az üzeneteket. A puffertől szinkron blokkolva várakozással kapjuk meg az üzeneteket.

Kapcsolódás a RabbitMQ –hoz és csatorna felépítés

Megegyezik a termelőnél leírtakkal.

Üzenetsor létrehozás

Megegyezik a termelőnél leírtakkal. Fontos, hogy az üzenetsor neve azonos legyen a termelőnél használttal. (QUEUE_NAME)

Puffer objektum létrehozása

```
    QueueingConsumer consumer = new QueueingConsumer(channel);
    channel.basicConsume(QUEUE_NAME, true, consumer);
```

Először létrehozzuk a puffer objektumot az adott csatornán. (QueueingConsumer) Után hozzárendeljük a puffer objektumot mint üzenetsor fogyasztót aszinkron üzenetküldési

célpontként az üzenetsorunkhoz. (basicConsume) Fontos, hogy a csatorna név megfelelő legyen. (QUEUE_NAME)

Üzenet fogadása

```
QueueingConsumer.Delivery delivery = consumer.nextDelivery();
String message = new String(delivery.getBody());
```

Az üzenetek szinkron blokkolva fogadásához a puffer objektum nextDelivery metódusa használható. Miután kinyertük az üzenet bájt tömb tartalmát a getBody() metódussal, ezt deszerializálni szükséges a megfelelő objektummá.

Csatorna és kapcsolat lezárása

Megegyezik a termelőnél leírtakkal.

Felkészülés a mérésre

Az elméleti felkészüléshez a AMQP leírása használandó, míg a gyakorlati felkészüléshez a RabbitMQ gyorstalpaló. A felkészüléshez javasolt először a RabbitMQ gyorstalpalót végignézni, ami gyakorlati példákon, és futtatható kódok mentén magyarázza el az alapokat. **Az ellenőrző kérdések részben a gyorstalpalóban megemlített problémás esetekre kérdeznek rá.**

Iránymutató az ellenőrző kérdésekhez

Az ellenőrző kérdések az AMQP leírás és a gyorstalpaló alapján készültek.

Üzenet alapú kommunikáció

Mi egy üzenet alapú kommunikáció alrendszer célja?

Mik egy üzenet alapú kommunikáció alrendszer feladatai?

Miben különbözik az üzenet alapú kommunikáció egy közvetlen hívástól? Milyen következményekkel lehet számolni?

Alapfogalmak

Átfogó kérdés: Ismertessen egy adott alapfogalmat:

Az üzenet alapú kommunikációban hol helyezkedik el?

Milyen részfeladatot lát el az üzenet alapú kommunikációban?

Milyen más fogalmakkal van kapcsolatban és miért?

Hogyan működik?

Exchange

Binding (Routes)

Queue

Publisher

Consumer

Binding key (*,#)

Routing_key

Exchange módok

Átfogó kérdés: Hogy működik és mire lehet használni egy exchange módot?

Direct exchange

Fanout exchange

Topic exchange

Headers exchange

Topic exchange esetén hányszor lesz kézbesítve az az üzenet, ami több mintára is illeszkedik?

Topic exchangeből hogyan lehet direct/fanout exchanget csinálni?

Hogy lehet direct exchangeből fanouthoz hasonlót csinálni? Mi lesz a különbség?

A direct exchange routolásának mi a megszorítása és melyik exchangekkel lehet feloldani ezt a megszorítást?

Mi történhet a nem továbbítható üzenetekkel?

Topic exchange illeszkedés

routing key	binding key	Illeszkedik?
””	””	
””	”#”	
””	”*”	
”.”	”*”	
”.”	”#”	
”..”	”*.*”	
”.a.”	”.*.”	

Fontos, hogy a . (pont) az üres szónak számít, így rá nem illeszkedik a * (csillag), de a # (kettős kereszt) igen.

Alapértelmezett beállítások

Mi az alapértelmezett exchange típus és hogyan kapcsolódnak hozzá a csatornák?

Mi az alapértelmezett csatorna kézbesítési mód, ha több fogyasztó is tartozik egy csatornához? Milyen problémát okozhat ez a mód? Hogyan lehet változtatni a módon?

Lehet e alapértelmezett beállításoknál a RabbitMQ-hoz bárhonnan csatlakozni?

Megbízható kommunikáció

Mi ellen véd az ack?(Message acknowledgment)

Mikor történik az üzenetek újraküldés? Mi jelzi a feldolgozó meghibásodását?

Mi az auto ack, mire használható és mire nem?

Milyen jelenség várható akkor, ha lemarad a manuális visszajelzés?

Mivel lehet egyszerűen védekezni a RabbitMQ szerver hibái ellen? Milyen egyéb megoldások vannak?

Mit állít vissza (mit biztosít) a queue durability és a message persistence?

Egyéb

Mikor és miért van szükség correlationId-ra? Milyen értékeket célszerű használni?

Milyen versenyhelyzet alakulhat ki, ami miatt az ismeretlen correlationID-val rendelkező üzeneteket célszerű eldobni?

Mi a Channel lényege? (Mivel takarékoskodik?)