Message Queue Fundamentals in .NET

Message Queue Proof-of-Concept

Elton Stoneman
geekswithblogs.net/eltonstoneman
@EltonStoneman

pluralsight
hardcore dev and IT training

Message Queue Proof-of-Concept

Replace synchronous processing with **fire-and-forget**

Using **MSMQ** for messaging

Verify messaging **works**, is **reliable** and **scalable**

Demo 1: Fire-and-Forget

Feature

Responsive unsubscribe user from mailing lists

Task

Remove unsubscribe in web & replace with message send

Task

Handle unsubscribe message in separate process

Demo 1: Fire-and-Forget

Demo 1: Fire-and-Forget

- **Create Unsubscribe command message**

```
var unsubscribeCommand = new UnsubscribeCommand
{
    EmailAddress = emailAddress
};
```

- **Send Unsubscribe message to queue**
  - From presentation layer (client)

```
using (var queue = new msmq.MessageQueue(".\\private$\\sixeyed..."))
{
    var message = new msmq.Message();
    var jsonMessage = JsonConvert.SerializeObject(unsubscribeCommand);
    message.BodyStream = new MemoryStream(
                                    Encoding.Default.GetBytes(jsonMessage));
    queue.Send(message);
}
```

Demo 1: Fire-and-Forget

- **Read Unsubscribe command message**
  - In console app (message handler)

```
using (var queue = new msmq.MessageQueue(".\\private$\\sixeyed..."))
{
   //...
   var message = queue.Receive();
   var reader = new StreamReader(message.BodyStream);
   var jsonMessage = reader.ReadToEnd();
   var unsubscribeMessage = JsonConvert.DeserializeObject
                                      <UnsubscribeCommand>(jsonMessage);
```

- **Handle message**

```
var workflow = new UnsubscribeWorkflow(unsubscribeMessage.EmailAddress);
workflow.Run();
```

Fire-and-forget

Fire-and-forget

Client builds and sends message
Message type, serialization format
Queue client, destination address

Queue stores message
Durable, sequential storage

Handler reads message
Queue client, source address
Serialization format, message type

Demo 2: Reliability

Feature

Ensure unsubscribe requests are always processed

Task

Verify messages are stored if there are no handlers listening

Task

Verify messages not lost if handler fails

Demo 2: Reliability

Reliability

Queue is **durable**, messages persisted until retrieved

Queue is **sequential**, messages retrieved in sent order

Queue is **not reliable**, messages can be lost

Demo 2: Reliability

- **Send Unsubscribe message to transactional queue**
  - From presentation layer (client)
  - With **MessageQueueTransaction**

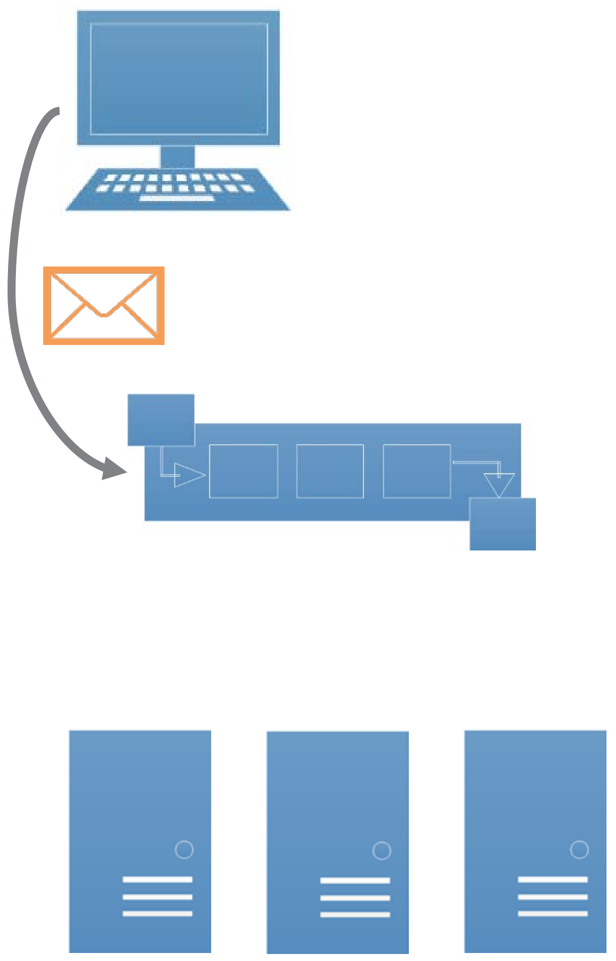
```
using (var queue = new msmq.MessageQueue(".\\private$\\sixeyed...-tx"))
{
    var message = new msmq.Message();
    var jsonMessage = JsonConvert.SerializeObject(unsubscribeCommand);
    message.BodyStream = new MemoryStream(
                                 Encoding.Default.GetBytes(jsonMessage));
    var tx = new msmq.MessageQueueTransaction();
    tx.Begin();
    queue.Send(message, tx);
    tx.Commit();
}
```

Demo 2: Reliability

- **Read Unsubscribe command message from transactional queue**
  - In console app (message handler)
  - With **MessageQueueTransaction**

```
using (var queue = new msmq.MessageQueue(".\\private$\\sixeyed..."))
{
    //
    using (var tx = new msmq.MessageQueueTransaction())
    {
        tx.Begin();
        var message = queue.Receive(tx);
        var reader = new StreamReader(message.BodyStream);
        var jsonMessage = reader.ReadToEnd();
        //...
        tx.Commit();
```

Performance & Scalability



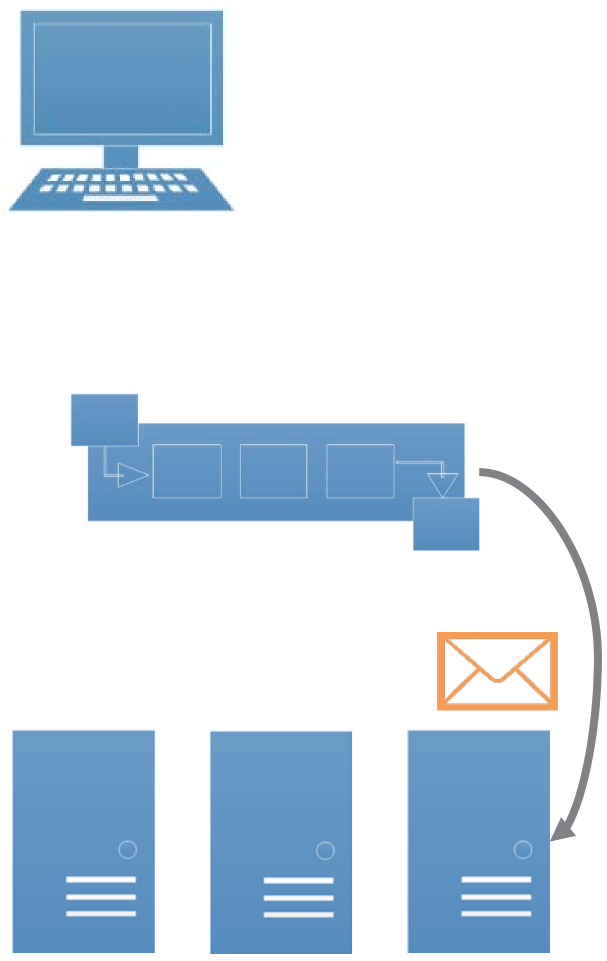
Client sends message to queue

Multiple times

Queue stores messages

In order received

Performance & Scalability

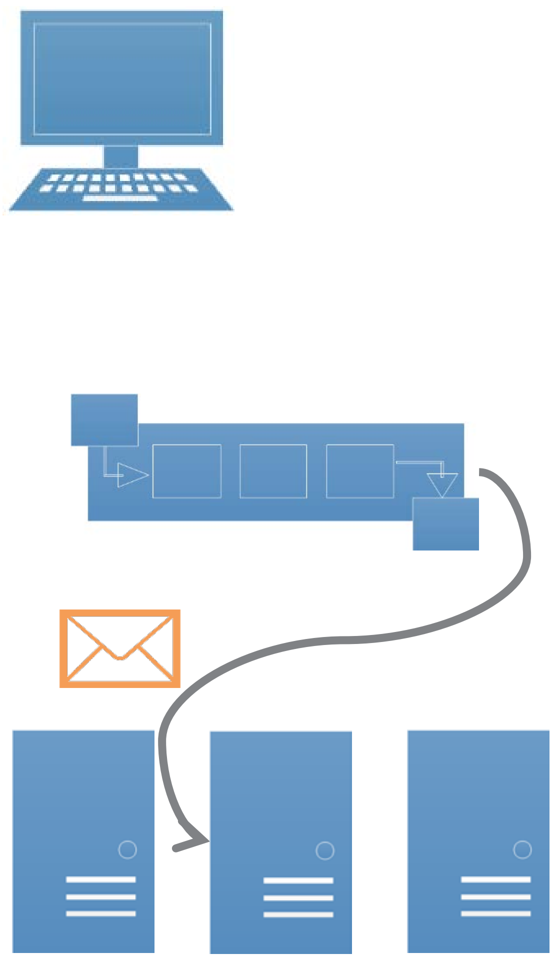


Client sends message to queue
Multiple times

Queue stores messages
In order received

Handler retrieves message
Any handler which has capacity

Performance & Scalability

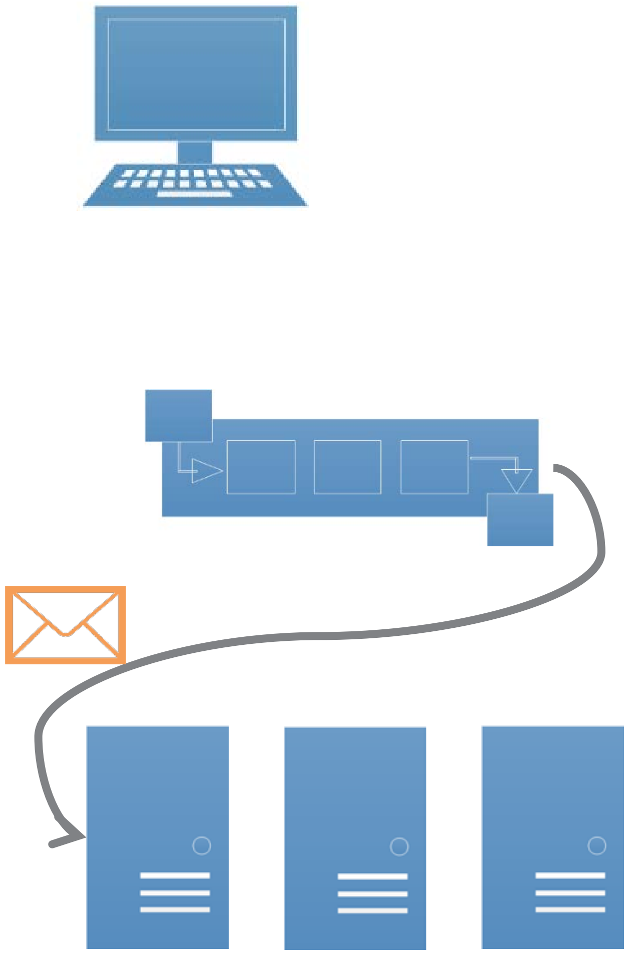


Client sends message to queue
Multiple times

Queue stores messages
In order received

Handler retrieves message
Any handler which has capacity

Performance & Scalability



Client sends message to queue

Multiple times

Queue stores messages

In order received

Handler retrieves message

Any handler which has capacity

Demo 3: Performance & Scalability

Feature

Ensure unsubscribe process can handle heavy loads

Task

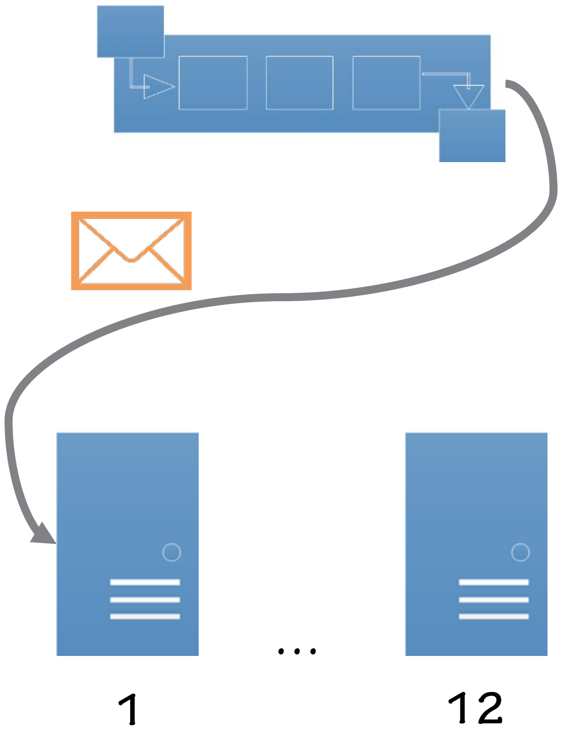
Verify messages are distributed between handlers

Task

Load test to compare synchronous and async processes

Demo 3: Performance & Scalability

Demo 3: Performance & Scalability



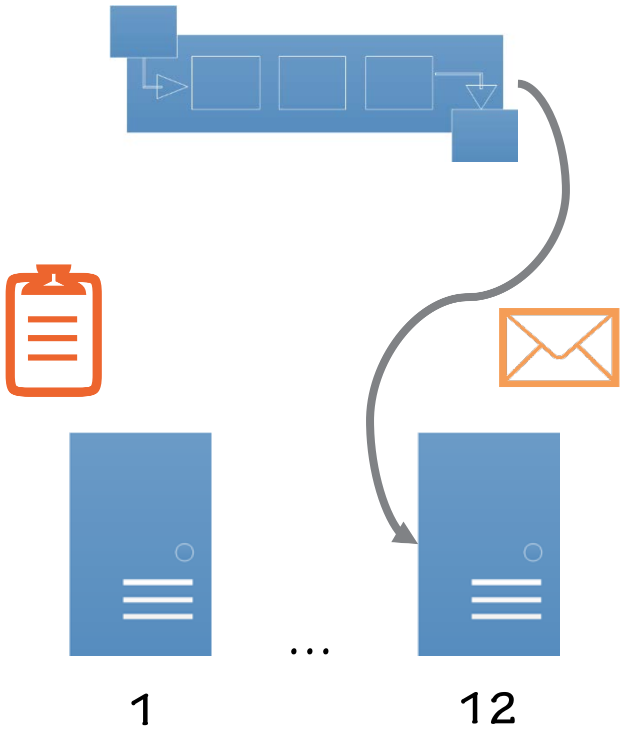
Populated queue

Multiple messages

Multiple message handlers

Retrieve messages sequentially

Demo 3: Performance & Scalability



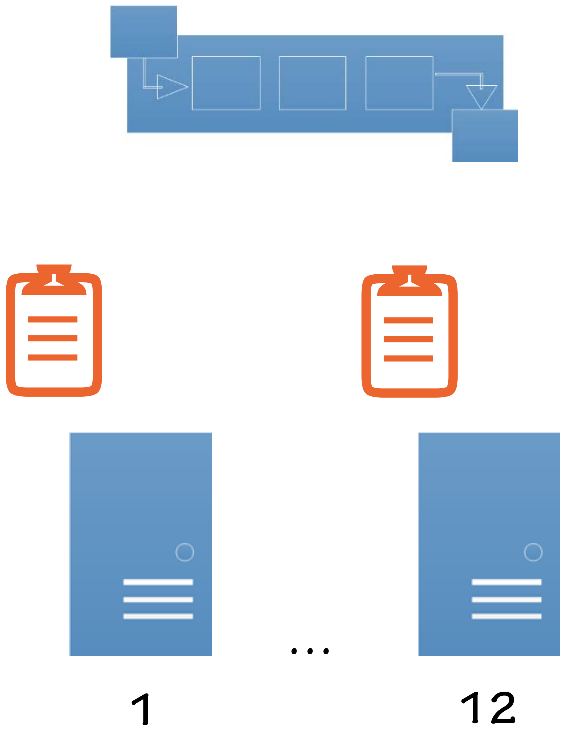
Populated queue

Multiple messages

Multiple message handlers

Retrieve messages sequentially

Demo 3: Performance & Scalability

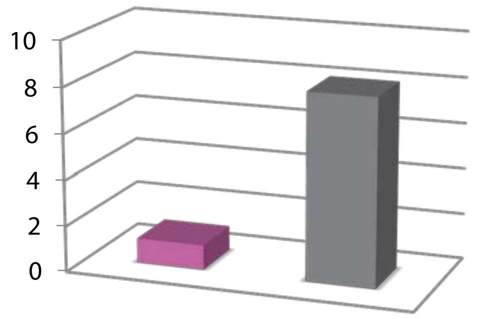


Populated queue
Multiple messages

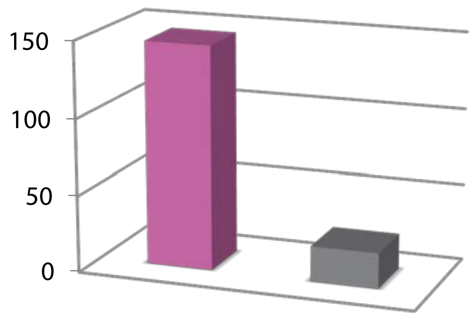
Multiple message handlers
Retrieve messages sequentially

Multiple messages in-flight
Distributed, concurrent processing

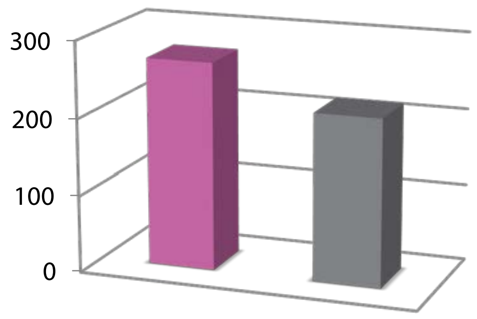
Demo 3: Performance & Scalability



Page **response time**
(smaller is better)



Web server **capacity**
(bigger is better)



Total **processing time**
(smaller is better)

🟪 - message queue

⬛ - synchronous call

Summary

- **Fire-and-forget PoC** ✓
  - Using MSMQ
- **Additional components** ✓
  - Message queue
  - Message handler (console)
  - Message definitions
- **Performance, scalability & reliability** ✓
  - Immediate user response
  - Concurrent, distributed processing
  - Durable, transactional messaging