YTÜ FEN EDEBİYAT FAKÜLTESİ

# MATEMATİK BÖLÜMÜ

## YAZILIM VE KUYRUK SİSTEMLERİ ARASINDA İLETİŞİMİ SAĞLAYAN MASSTRANSİT VE NSERVİCEBUS UYGULAMALARININ KARŞILAŞTIRILMASI

**Nur Sena KAYA 18025048**

**Prof.Dr Adem Cengiz ÇEVİKEL**

Bu çalışmada yazılım uygulamaları ve veri aktarımını sağlayan mesajlaşma sistemleri arasındaki iletişimi ve bu iletişimi sağlayan NServiveBus ve MassTransit uygulamalarının özellikleri araştırılmış olup çeşitli yönlerde bu iki uygulamanın karşılaştırması yapılmıştır.

### YAZILIM

Yazılım, belirli görevleri gerçekleştirmek için tasarlanmış veri koleksiyonu, kurallar ve talimatların bir bütünüdür. Bu talimatlar, bilgisayarın nasıl çalıştığını, kullanıcıların neler yapabileceğini ve cihazların bu işlemlere nasıl tepki vereceğini belirler. Yazılım, fiziksel bileşenler olan donanımın kullanılabilir ve işlevsel hale gelmesini sağlamak için kodlama dilleri kullanılarak yazılıp geliştirilen bir bütündür.

- SİSTEM YAZILIMI
- UYGULAMA YAZILIMI
- PROGRAMLAMA YAZILIMI
- AĞ YAZILIMI
- GÖMÜLÜ SİSTEM YAZILIMI
- WEB YAZILIMI VE HİZMETLERİ

### Asenkron İletişim

Bir işlemin başlatılmasıyla tamamlanması arasında zaman bağımsızlığına dayanan bir iletişim modelidir. İki uygulama arasında gerçekleşen iletişimde, gönderen uygulama işlemi başlatır ve sonucunu beklemeden diğer işlemlere geçebilir. Alıcı uygulama ise, işlemi uygun bir zamanda alır ve sonuçlandırır. Bu model, esneklik ve ölçeklenebilirlik avantajları sağlar.

### İletişim Sağlamak İçin İzlenecek Adımlar

1. İletişim Sağlamak İçin İzlenecek Adımlar
2. Kuyruk Sistemi Seçimi
3. Sistem Entegrasyonu
4. Mesaj Formatı ve Protokollerin Belirlenmesi
5. Güvenlik Ayarlarının Yapılması
6. Hata Yönetimi ve İzleme
7. Test ve Doğrulama
8. Bakım ve Optimizasyon
9. Dokümantasyon

### MESAJLAŞMA ÇERÇEVELERİ

- Apache Camel
- Akka.Net
- NServiceBus
- MassTransit
- Rebus
- Spring Integration

### Mesaj Taşıma Mekanizmaları

Güvenilir ve etkili bir iletişimi destekleyerek uygulamalar arasındaki mesajlaşmayı sağlar.

- RabbitMQ
- Azure Service Bus
- MSMQ
- SQL Server
- Apache Kafka
- NATS
- Zero MQ
- Active MQ
- IBM MQ
- Redis

### Avantajları

- Asenkron iletişimi destekleyerek esnek ve ölçeklenebilir mimariler sağlar.
- Çoklu iş parçacığı veya işlem desteği ile iş yükünü etkili bir şekilde dengeleyerek kaynakları kullanır.
- Beklenmeyen yük artışlarına uyum sağlayarak sistemi düzenler.
- Güvenilirlik ve dayanıklılık özellikleri ile veri kaybını önler.
- Farklı bileşenleri kolayca entegre edebilir, modüler sistem mimarilerini destekler.
- Tüketici tarafında hata toleransı ile başarısız işlemleri ele alır.
- İş yükünü dengeleyerek öncelikli görevleri belirli bir sırayla işler.
- Çeşitli uygulama alanlarında kullanılır, mesaj sıralama, görev sıralama, asenkron işleme ve dağıtık sistemlerde avantajlar sağlar.

### Dikkat Edilmei Gereken Noktalar

- Kuyruk sistemleri performans ve ölçeklenebilirlik için tasarlanmalıdır.
- Etkili hata yönetimi stratejisi belirlenmeli, veri güvenliği sağlanmalıdır.
- Öncelik atama, zamanlama ve kaynak yönetimi önemlidir.
- İşlem sırasında beklenen durumlar planlanmalı, hata toparlama mekanizmaları bulunmalıdır.
- Performans izleme ve sorun tespiti için araçlar kullanılmalıdır.
- Çoklu işlemcili ortamlarda senkronizasyon dikkatle yönetilmelidir.
- Sistemler arası entegrasyon düzgün ve güvenilir olmalıdır.

### KUYRUK SİSTEMLERİ

Kuyruk sistemi, verilerin belirli bir sırayla işlendiği ve geçici olarak depolandığı bir yapıdır. Bu sistem, "First In First Out" (FIFO) prensibiyle çalışarak ilk eklenen verinin ilk çıkarıldığı bir düzeni benimser. Bilgisayar bilimleri ve bilişim sistemlerinde sıkça kullanılan bu sistem, çoklu işlemcili sistemlerde, ağ iletişiminde, iş parçacığı programlamasında ve asenkron veri işleme durumlarında etkili bir şekilde kullanılır. Kuyruk sistemleri, verilerin düzenli bir şekilde işlenmesini, sırayla beklemesini ve sistemlerin daha etkili, düzenli ve ölçeklenebilir bir şekilde çalışmasını sağlar. · Kuyruk (Queue): Verilerin veya işlemlerin depolandığı yapı.

· Ekleme (Enqueue): Kuyruğa yeni bir eleman eklenir. Bu işlem genellikle kuyruğun sonunda gerçekleşir.
· Çıkarma (Dequeue): Kuyruğun başındaki eleman çıkarılır. Bu işlem, kuyruktaki elemanların işlenmesi için kullanılır.
· FIFO Prensibi: İlk gelen veri, ilk çıkarılır. Bu prensip, kuyruk sisteminin temel işleyiş prensibidir.
· Kuyruk Boyutu: Kuyruğun kaç eleman içerdiğini belirten bir boyut sınırlaması olabilir. Bu, kuyruğun ne kadar dolu veya boş olduğunu kontrol etmek için kullanılabilir.

### Kuyruk Mimarileri

- Tekli Kuyruk
- Öncelikli Kuyruk
- Çoklu Kuyruk

### Kuyruk Yönetimi

- İzleme ve Analiz
- Performans Optimizasyonu
- Ölçeklenebilirlik Öncelik Yönetimi
- Bloklayan ve Blok Yok Mekanizmaları
- Hata Yönetimi İş Parçacığı veya İşlem Yönetimi
- Kaynak Yönetimi

### Uygulama Alanları

- Bilgi İşlem ve Algoritmalar
- Bilgi İletişimi
- Arka Plan İşlemler
- Dağıtık Sistemler

### NServiceBus

- NServiceBus, mesajlar aracılığıyla iletişim kurarak bağımsız geliştirme ve dağıtımı kolaylaştırır.
- Güvenilir iletim mekanizmalarıyla mesajların kaybolmasını önler, ölçeklenebilir ve hata toleranslıdır.
- Yüksek taleplerde ölçeklenebilir, işlenemeyen mesajları otomatik yönetir.
- Farklı iletişim altyapılarına uyumlu, sagalar ve olay tabanlı mimarilere destek sağlar.
- Yapılandırılabilir, genişletilebilir ve çeşitli veritabanlarıyla uyumludur.

Çalışma Akışı

1. Mesaj Oluşturma ve Gönderme
2. Mesaj Taşıma
3. Mesaj İşleme
4. Hata Yönetimi
5. Ölçeklenebilirlik ve Yük Dengeleme
6. İleri Seviye İşlemler
7. İşlem Sonu

### MassTransit

- MassTransit, yayıncı/tüketici deseniyle mesajları yayınlama ve tüketme desenlerini destekler, servisler arası etkileşimi sağlar.
- Gelişmiş mesajlaşma özelliklerine sahiptir, mesaj yeniden deneme politikaları, zamanlanmış mesajlar ve işlem garantileri gibi özellikleri destekler.
- Çeşitli mesaj kuyruğu sistemleriyle entegre olabilir, RabbitMQ, Azure Service Bus, Amazon SQS gibi altyapılara esnek bağlantı imkanı sunar.
- SAGA deseni desteği ile uzun süreçli işlemleri koordine etmek için kullanılır.
- Asenkron işleme modelini destekler, yanıt sürelerini artırır ve kaynak kullanımını optimize eder.
- Yüksek ölçeklenebilirlik ve performans için uygun bir çözüm sunar, yüksek trafikli uygulamalara hitap eder.

Çalışma Akışı

1. Mesaj Tanımlama
2. Endpoint Yapılandırması
3. Mesaj Gönderimi
4. Mesaj Alımı ve İşlenmesi
5. Sagalar ve Uzun Süreli İş Akışları
6. Mesaj Yönlendirme
7. Hata Yönetimi
8. İzleme ve Günlükleme

### NSERVICEBUS & MASSTRANSIT

**Ortak Özellikleri**

- .NET Framework veya .NET Core Gereksinimi
- Mesaj Protokolleri
- Gevşek Bağlantılı Mimari
- Ölçeklenebilirlik
- Taşıyıcı Çeşitliliği
- Saga Desteği
- Gelişmiş Hata Yönetimi
- Kullanıcı ve Geliştirici Topluluğu
- Geliştirme Ortamı
- Bağımlılık Yönetimi

**Farklı Özellikleri**

- Lisanslama
- Yapı ve Yaklaşım
- Özellik Seti
- Ölçeklenebilirlik
- Destek ve Topluluk
- Kullanım Kolaylığı

**Performans Test Sonuçları**

- Tekli gönderim

1: NServiceBus avg 45ms
MassTransit avg 30ms
100-5: NserviceBus avg 316ms
MassTransit avg 589ms
1000-5: NServiceBus avg 260ms
MassTransit avg 489ms

- Çoklu gönderim

250-1: NServiceBus avg 760ms
MassTransit avg 541ms
500-1: NServiceBus avg 777ms
MassTransit avg 1081ms
500-10: NServiceBus avg 6500ms
Masstransit avg 6000ms
500-20: NServiceBus avg 10000ms
Masstransit avg 10000ms

- **MassTransit - ortalama 500ms**
- **NServiceBus - ortalama 31ms**

### Karşılaştırma Sonuçları

- MassTransit ücretsiz ve açık kaynak, NServiceBus ise maliyetli ve lisanslıdır.
- MassTransit çeşitlilik ve esneklik sağlarken, NServiceBus Microsoft ürünleriyle derin entegrasyon sunar
- MassTransit, geniş broker seçenekleri ve esnek yapılandırma ile öne çıkarken, NServiceBus güçlü hata yönetimi ve ölçeklenme stratejileri sunar
- MassTransit, açık kaynak ve basit API yapısıyla geliştiricilere kolay erişilebilirlik sağlarken, NServiceBus daha geniş özellik seti ve ticari destek odaklı avantajlar sunar
  NServiceBus daha güçlü hata yönetimi ve sistem dayanıklılığına odaklanırken, MassTransit çeşitlilik ve esneklik avantajlarına sahiptir
- NServiceBus, gelişmiş görsel araçlar ve entegrasyonlar sunarak monitoring ve debugging konusunda öne çıkarken, MassTransit açık kaynak esnekliği ve geniş entegrasyon avantajlarıyla dikkat çeker.
- MassTransit, geniş broker desteği ve açık kaynak esnekliğiyle öne çıkarken, NServiceBus özellikle Microsoft entegrasyonları ve kurumsal destek sağlamasıyla dikkat çeker, tercih projenin ihtiyaçlarına bağlıdır.
- MassTransit geniş broker desteği ve esnek yapılandırma ile öne çıkarken, NServiceBus daha kapsamlı Saga yönetimi ve kurumsal düzeyde sistem karmaşıklığını yönetme yetenekleriyle dikkat çeker
- MassTransit esnek yapılandırma ve geniş broker desteği sunarken, NServiceBus kapsamlı Saga yönetimi ve kurumsal düzeyde sistem karmaşıklığını yönetme yetenekleriyle öne çıkar.

KAYNAKÇA
[1] MassTransit · MassTransit
[2] NServiceBus • Particular Docs
[3] Message Queues | System Design - GeeksforGeeks