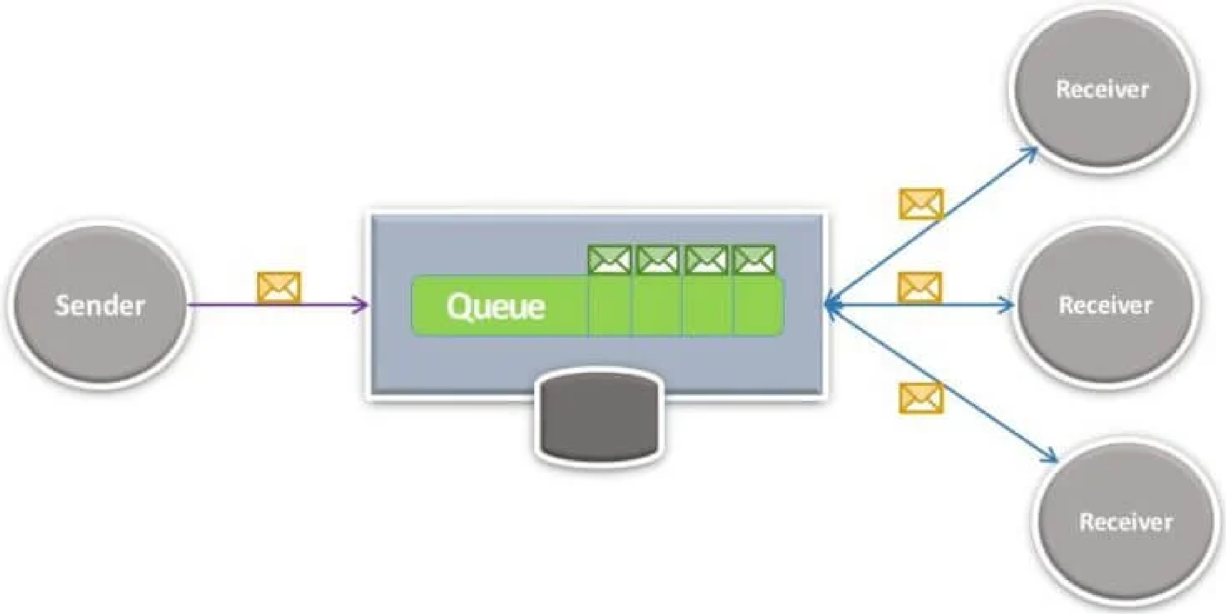


Message Queues & You – 12 Reasons to Use Message Queuing

MATT WATSON | JANUARY 30, 2017 |
DEVELOPER TIPS, TRICKS & RESOURCES (HTTPS://STACKIFY.COM/DEVELOPERS/)

At Stackify, we receive a lot of data from *your apps* to our Retrace APIs. We queue all of that data as soon as it gets to us, and then we use separate background services to process the data. Message queues help a lot with ensuring data is never lost, traffic spikes, etc. We decided to put together a list of why you should be using message queues (/what-is-csharp-queue/)!

1. Redundancy via Persistence

Redundancy is one of the most obvious advantages to message queues. Application crashes, timeouts, errors in your code, and other problems are just part of the norm. This is especially true in applications that process millions or billions of transactions per month.

Queues help with redundancy by making the process that reads the message confirm that it completed the transaction and it is safe to remove it. If anything fails, worst case scenario, the message is persisted to storage somewhere and won't be lost. It can be reprocessed later.

2. Traffic Spikes

You don't always know exactly how much traffic your application is going to have. For example, at Stackify we receive billions of messages a month. We have no way to know what our clients are going to send us. By queuing the data we can be assured the data will be persisted and then be processed eventually, even if that means it takes a little longer than usual due to a high traffic spike.

3. Improve Web Application Page Load Times

Queues can be useful in web applications to do complex logic in a background thread so the request can finish for the user quickly. If someone places an order on your website, that could involve a lot of different things that have to happen. You can do the minimum and return success to your user and kick off the rest of them in a background thread to finish up, without using a full message queuing system and background apps. Most programming languages have ways to do this now. Examples: Resque (https://github.com/resque/resque), Hangfire (http://hangfire.io), etc.

4. Batching for Efficiency

Batching is a great reason to use message queues. It is much more efficient to insert 100 records into a database at a time instead of 1 at a time, 100 times. We insert a lot of data into elasticsearch and SQL Server. Batching helps us optimize their performance by tuning the size of the transactions.

5. Asynchronous Messaging

Queues can be great in scenarios where your application needs something done but doesn't need it done now, or doesn't even care about the result. Instead of calling a web service and waiting for it to complete, you can write the message to a queue and let the same business logic happen later. Queues are an excellent way to implement an asynchronous programming (https://stackify.com/asynchronous-programming-easier-think/) pattern.

6. Decouple by Using Data Contracts

By using a queue between different parts of your software you can decouple the hard dependencies. The format of the message in the queue becomes your data contract and anything that knows how to read that message format can be used to process the transaction. This could be useful for parts of your code that are even written in different programming languages.

7. Transaction Ordering and Concurrency Challenges

If 1000 people are placing an order on your website at one time, that could create some problems with concurrency and ensuring that the first order in finishes first. By queuing them up, you could then guarantee their order and control how many are even processed concurrently.

8. Improve Scalability

Message queues enable you to decouple different parts of your application and then scale them independently. Using Azure, AWS, or other hosting solutions you could even dynamically scale that background service based on CPU usage or other metrics. Message queues can help a lot with scalability and elasticity.

9. Create Resiliency

By breaking your app up and separating different components by queues, you inherently create more resiliency. Your website can still be operational even if part of the back end processing of the order is slightly delayed. At Stackify, we design our incoming APIs to do as little as possible beyond queuing the data to ensure that nothing else can bring down the ability to receive data. Even if SQL Server is down, we want to be able accept data!

10. Guarantee Transaction Occurs Once

Message queues are designed to be transactional. When you read a message off of a queue, you have to tell it where you completed processing the message before it is completely removed from the queue. This helps to ensure that a transaction only occurs once.

11. Break Larger Tasks into Many Smaller Ones

Another good use for queues is breaking up a larger task into lots of little pieces and then queuing them all up. We have a good example of this at Stackify. If you edit a monitoring template for a group of servers, we then need to update the monitors on every server that uses that template. We can queue up a message for every server and potentially do them concurrently as smaller operations.

12. Monitoring

Message queuing systems enable you to monitor how many items are in a queue, the rate of processing messages, and other stats. This can be very helpful from an application monitoring standpoint to keep an eye on how data is flowing through your system and if it is getting backed up.

Does every app need message queues?

No, of course not. Like most things in life, there is a place for everything.

However, I would suggest at least thinking about improving the performance of your web applications by using background tasks to offload some long running transactions.

Alternative “In app” Background Processing

You can consider Hangfire (http://hangfire.io/), Resque and other similar projects for this. Those projects act as sort of mini queuing systems and can persist the jobs for resiliency. They run entirely in your web application so it doesn’t require standing up additional servers or anything.

Another option without the resiliency is using fire and forget tasks in your code. But if your app crashes before it finishes or some sort of error occurs, that operation would be lost.

ASP.NET has a handy way to do this. If you do it this way, the framework will even delay stopping your app until your code finishes. But if your app crashes, it would be lost. Using Hangfire would provide more functionality and the resiliency.

```
HostingEnvironment.QueueBackgroundWorkItem(x =>
{
          //do some stuff here, like maybe send an email?
          SmtpClient smtp = new SmtpClient();
          smtp.Send();
});
```

You can read more about HostingEnvironment.QueueBackgroundWorkItem at MSDN (https://blogs.msdn.microsoft.com/webdev/2014/06/04/queuebackgroundw to-reliably-schedule-and-run-background-processes-in-asp-net/).

Message Queues Conclusion

Message queues are a very critical component to our platform at Stackify. We queue and process billions of messages a month. For us, it would be hard to imagine not having message queues!

It can be a lot of work to stand up RabbitMQ, MSMQ or some other message queuing service. Although, Azure and AWS make it easier with their hosted offerings. We use Azure Service Bus ourselves and it has worked well for us.

How do you use message queues? Have any other great uses or tips? Comment below!

Improve Your Code with Retrace APM

Stackify's APM tools are used by thousands of .NET, Java, PHP, Node.js, Python, & Ruby developers all over the world.
Explore Retrace's product features to learn more.

(/retrace-application-performance-management/)

App Performance Management (https://stackify.com/retrace-application-performance-management/)

(/retrace-code-profiling/)

Code Profiling (https://stackify.com/retrace-code-profiling/)

(/retrace-error-monitoring/)

Error Tracking (https://stackify.com/retrace-error-monitoring/)

(/retrace-log-management/)

Centralized Logging (https://stackify.com/retrace-log-management/)

(/retrace-app-metrics/)

App & Server Metrics (https://stackify.com/retrace-app-metrics/)

Learn More
(/retrace/)

About the Author | Latest Posts

About Matt Watson

Matt is the Founder & CEO of Stackify. He has been a developer/hacker for over 15 years and loves solving hard problems with code. While working in IT management he realized how much of his time was wasted trying to put out production fires without the right tools. He founded Stackify in 2012 to create an easy to use set of tools for developers.

(https://twitter.com/mattwatson81) in
(https://www.linkedin.com/in/mattwatsonkc/)

Email

Search Stackify

Search

Topics/Keywords

ASP.NET (https://stackify.com/?tag=asp.net,net-core)

Product Updates (https://stackify.com/stackify/)

.NET Core (https://stackify.com/content/net-core/)

App Monitoring (https://stackify.com/?tag=monitoring,apm)

Java (https://stackify.com/content/java/)

App Performance Tips (https://stackify.com/?tag=performance,profiler,apm)

Azure (https://stackify.com/content/azure/)

Error Handling (https://stackify.com/?tag=exception,exceptions,error,errors)

AWS (https://stackify.com/content/AWS/)

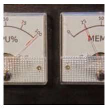
Logging Tips (https://stackify.com/?tag=logs,logging)

Cloud (https://stackify.com/?tag=cloud,azure,aws)

DevOps (https://stackify.com/content/DevOps/)

Popular Posts

ASP.NET Performance: 9 Types of Tools You Need to Know!



How to Troubleshoot IIS Worker Process (w3wp) High CPU Usage

How to Monitor IIS Performance: From the Basics to Advanced IIS Performance Monitoring

SQL Performance Tuning: 7 Practical Tips for Developers

Looking for New Relic Alternatives & Competitors? Learn Why Developers Pick Retrace

Recent Posts

(https://stackify.com/how-to-monitor-your-web-application-availability/) How to monitor your web application availability (https://stackify.com/how-to-monitor-your-web-application-availability/)

(https://stackify.com/netreo-expands-apm-capabilities-with-strategic-acquisition-of-stackify/)

Netreo Expands APM Capabilities with Strategic Acquisition of Stackify (https://stackify.com/netreo-expands-apm-capabilities-with-strategic-acquisition-of-stackify/)

(https://stackify.com/stackify-netreo-blog/) Stackify + Netreo Creates a Dev + Ops Powerhouse (https://stackify.com/stackify-netreo-blog/)

(https://stackify.com/how-to-optimize-js-performance/) How to Optimize JavaScript Performance (https://stackify.com/how-to-optimize-js-performance/)

(https://stackify.com/ultimate-guide-performance-testing-and-software-testing/) The Ultimate Guide to Performance Testing and Software Testing: Testing Types, Performance Testing Steps, Best Practices, and More (https://stackify.com/ultimate-guide-performance-testing-and-software-testing/)

(/stackify-developer-ebooks/)

ckify.com/) (https://stackify.com/guest-blogging-guidelines/)

Get In Touch

Contact Us

Request a Demo

Start Free Trial

Products

Retrace

Prefix

.NET Monitoring

Java Monitoring

PHP Monitoring

Node.js Monitoring

Ruby Monitoring

Python Monitoring

Retrace vs New Relic

Retrace vs Application Insights

Solutions

Application Performance Management

Centralized Logging

Code Profiling

Error Tracking

Application & Server Monitoring

Real User Monitoring

For Developers

For DevOps

Resources

What is APM?

Pricing

Case Studies

Blog

Documentation

Free eBooks

Free Webinars

Videos

Ideas Portal

ROI Calculator

Support

News

Comp

(https://stackify.com/)

About Us

News

Careers

GDPR

Security Information

Terms & Conditions

Privacy Policy

PO Box 2159
Mission, KS 66201
816-888-5055 (tel:18168885055)

(https://www.facebook.com/Stackify/)

(https://twitter.com/Stackify)

(https://www.youtube.com/channel/UCnxYNQDtTr_ZYUnMjeACfCg)

(https://www.linkedin.com/company/stackify/)