CA2 Research Project Proposal

Investigating the impact of configuration options upon the performance of Azure Service Bus

*Brian Skehan*

*X00205786@myTUDublin.ie*

# Abstract

**This paper introduces a proposal for a research project which investigates the impact of differing configuration options upon performance of Azure Service Bus in a software system using an Event Driven Architecture.**

# List of keywords

Event Driven Architecture, Azure Service Bus, Performance, Configuration

# Introduction

Event Driven Architecture (EDA) is a common pattern used when building applications where information flows between components of a software system by producing and consuming events. Events can be state changes or actions such as readings from sensors, system messages, user interactions etc. which could be of interest to other elements of the software system.

EDA can be used to develop a software system that requires high availability and high performance in addition to low latency (Khriji, et al., 2022)

Performance can be measured through a multitude of metrics, but Throughput, Response Times, Compute and memory usage would be common (Cabane & Farias, 2024) (Rahmatulloh, et al., 2022). These metrics should be measured in response to running time-bound experiments with known loads and configurations (Ezzeddine, et al., 6 Dec 2021) (Rahmatulloh, et al., 2022).

Configuration options on software systems allow for the systems to be customised and adapted to the needs the system and to enable the system to react to changing load or to respond to events.

Introducing EDA into a software system allows components to be isolated from each other and therefore more loosely coupled. Determining performance levels in such a system is more difficult due to the levels of abstraction. As such it becomes more important to understand how differing configurations of core components of the software system impact upon the performance levels of the system.

Within EDA an event or message handling system (Microsoft, 2024) is a core component to enable the pattern. Azure Service Bus is an example of a message handling system which can be used to implement an EDA.

# Objective

The objective of this research project is to investigate the impact of different configuration options within Azure Service Bus upon its performance for known loads of events within defined timespans.

The following research questions are posed:

RQ1: How do selected configuration options impact performance of Azure Service Bus?

RQ2: How does event or message payload size impact upon performance of Azure Service Bus?

RQ3: Does the number of subscriptions to an Event topic impact upon performance of Azure Service Bus?

These research questions can be answered by developing a producer application which will generate events/messages to be placed upon Azure Service Bus (ASB) queues which would then be consumed by a Consumer application. Analysing logs and data created by the applications and by ASB will enable the research questions to be answered.

# Methodology

* Create an instance of Azure Service Bus in an Azure region with a known configuration.
* Run one or more instances of the producer application on virtual machine(s) in the same region to generate significant load onto the ASB.
* Depending on the configuration to be tested connect one or more Consumer applications to the ASB.
* Preferably the test environment will be created and destroyed using Infrastructure as Code such as Bicep.
* Details of each run, its configuration along with the output logs will be recorded for analysis.

# Measuring Performance

Performance for an EDA can be measured through several metrics (Cabane & Farias, 2024). Some examples of relevant performance metrics are:

Throughput: Measuring the number of events processed per unit of time. High throughput is desirable in an EDA, indicating that the system can handle large volumes of events efficiently.

Latency: Time taken for an event to be received after it has been sent. Used to evaluate the responsiveness of the system. Low latency would be a crucial measure in real-time application.

Resource Utilization: CPU, memory, and network used by the message system within the EDA. Used to understand the capacity and scaling needs of the system.

Consumer Lag: the number of events that are waiting in the queue to be processed. Ideally zero but can identify where events are being produced faster than they can be consumed.

# Research Question detail.

## RQ1: How do selected configuration options impact performance?

Azure Service Bus has several configurable features which would be worth examining to investigate their impact upon performance:

To establish baseline performance an instance of Azure Service Bus will be created with all default options selected. The producer and consumer applications will then be deployed upon separate VMs within the same Azure region as the Service Bus instance. Multiple runs of the experiment would be run at different times of day and day of week to account for variation in infrastructure conditions with the Azure region.

For RQ1 further runs would be defined and executed with different combinations of options enabled or not on ASB. The ASB instance would need to be destroyed between runs as most of the options for configuration are only available upon creation of the ASB instance.

The following configuration options in Azure Service Bus will be considered for investigating their impact on performance of ASB.

### Duplicate Detection: Service Bus can detect Duplicate Messages (if enabled) by tracking the MessageId of all messages within a specified time window. If any new Message is sent with the same MessageId as one existing within the specified time window it will be dropped and ignored. The longer the time window the more MessageIds must be stored and compared against, hence it has an implication for the performance of the system.

### First In First Out guarantee: With potentially multiple concurrent receivers there is no guarantee that if Message A was sent before Message B that Message A will be processed before Message B. By enabling sessions on Azure Service Bus, messages can be sent within the context of a session and the first receiver processing a message from that session will retain a lock on that session and will therefore process in order all further messages for that session. The implication for performance is that only a single receiver can now process messages for that session even if other receivers are available.

### Partitioning: Enables queues within Azure Service bus to be partitioned across multiple message brokers. This would help isolate queues from issues being encountered on a single message broker. In Standard/Basic Tier cannot define the number of partitions but can enable partioning. In Premium tier the number of partitions can be defined at time of creation of the ASB instance. Performance would be impacted as once enabled there would be multiple message brokers processing the messages.

### Messaging Units (Premium Tier): The Premium Tier of Azure Service Bus provides for resource isolation and only workloads sent to the instance would be processed by the underlying resources. By default, premium tier allocates one (1) Messaging Unit, but this can be increased in increments (1, 2, 4, 8, 16). This can be scaled dynamically in reference to metrics such as CPU, throttling etc.

## RQ2: How does Event or Message payload size impact upon performance?

Azure Service Bus has a max message size of 256KB on Standard Tier or 100 MB on the Premium Tier. Message sizes of 1MB (Microsoft, 2024) can be achieved if using the SBMP protocol but as support for SBMP will be retired by Oct 2026 (Microsoft, 2023) it will be taken out of scope. There is also a max queue/topic size of 80 GB that should be taken into consideration as with a high number of events being placed onto a queue it may run out of storage space and so reject further events being placed upon the queue.

To establish baseline performance an instance of Azure Service Bus will be created with all default options selected. The producer and consumer applications will then be deployed upon separate VMs within the same Azure region as the Service Bus instance.

Multiple runs of the experiment would then be run at different times of day and day of week to account for variation in infrastructure conditions with the Azure region.

For RQ2 the Producer application will be configured on different runs to create messages with small payload size, random payload size between the minimum and maximum and runs with the maximum payload size.

## RQ3: Does the number of subscriptions to an Event impact upon performance?

Azure Service bus can operate as a simple queue with 1 or more competing consumers, or it can also be configured where the consumers subscribe to a Topic to which the Producer will publish events/messages.

To establish baseline performance an instance of Azure Service Bus will be created with all default options selected. The producer and consumer applications will then be deployed upon separate VMs within the same Azure region as the Service Bus instance. Multiple runs of the experiment would be run at different times of day and day of week to account for variation in infrastructure conditions with the Azure region.

Runs of the experiment will then be defined and executed where the Azure Service Bus operates as a simple queue, configured with Topics with 1 subscriber and other amounts of subscribers. Azure Service Bus has a limit of 2,000 subscribers per topic but would need to investigate an automated process for creating large numbers of subscribers so the variations of numbers of Consumers may just be limited to 1, 5 and 10 subscribers.

# Producer and Consumer applications

Consideration was given to running these load tests using a load test framework such as Apache JMeter or commercial load testing infrastructure such as LoadRunner or Blazemeter but the most efficient mechanism for creating events on Azure Service Bus is to use the AMQP protocol made available through clients provided by Microsoft for various languages. Native support within the load test tools appears limited and dependent on third party plugins which may add a level of unnecessary complication.

The test harness applications will therefore be written in .NET using C#. Where necessary configuration options will be provided to modify the operation of application and to indicate the Azure Service Bus instances to connect to. Log files will be written to disk in a manner so that they do not impact performance of the Producer or Consumer to any significant extent.

# Logs and Monitoring.

Producer and Consumer applications will output console and text logs with metrics. Correlation Ids will be set as part of the message so that metrics can be measured down to the individual message across the Producer, Consumer, and the Azure Service bus.

The logs will record the volumes of messages/events sent and received, the time of sending/receiving of any message. Azure Application Insights will also be used to provide further instrumentation.

Azure Service Bus will use Azure Monitor to assist in monitoring CPU, Memory, Queue Length, and any other metrics that might become interesting. The full list of metrics available on Azure Service Bus can be seen at <https://learn.microsoft.com/en-us/azure/service-bus-messaging/monitor-service-bus-reference>

To compare performance for each Research Question, the following charts or calculations may prove useful:

* Throughput figures for each run will be graphed to compare against each other on a Bar Chart
* Latency figures can also be calculated and graphed to compare performance for the varying options and configurations across the runs.
* Calculation and comparison of the average, median and 99th percentile for throughput and latency will give further insight into the base metrics.
* The Premium tier of Azure Service bus is billed quite differently and higher to the Basic and Standard tiers while promising higher performance. Calculating the approximate cost of each run will add perspective to what the performance / cost ratio would be.

# References

Cabane, H. & Farias, K., 2024. On the impact of event-driven architecture on performance: An exploratory study. *Future Generation Computer Systems,* Volume 153, pp. 52 - 69.

Ezzeddine, M., Tauvel, S., Baude, F. & Huer, F., 6 Dec 2021. *On The Design of SLA-Aware and Cost-Efficient Event Driven Microservices.* Virtual (Canada), Association for Computing Machinery.

Khriji, S., Benbelgacem, Y., Cheour, R. & El Houssaini, D., 2022. Design and implementation of a cloud‑based event‑driven architecture for real‑time data processing in wireless sensor networks. *The Journal of Supercomputing,* 78(3), pp. 3374 - 3401.

Microsoft, 2023. *Some Azure Service Bus SDK libraries will be retired on 30 September 2026—migrate to the latest SDKs.* [Online]   
Available at: https://azure.microsoft.com/en-us/updates/retirement-notice-update-your-azure-service-bus-sdk-libraries-by-30-september-2026/  
[Accessed 5 May 2024].

Microsoft, 2024. *Event-driven architecture style.* [Online]   
Available at: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/event-driven  
[Accessed 1 May 2024].

Microsoft, 2024. *Large messages support.* [Online]   
Available at: https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-premium-messaging#large-messages-support  
[Accessed 4 May 2024].

Rahmatulloh, A., Nugraha, F., Gunawan, R. & Darmawan, I., 2022. *Event-Driven Architecture to Improve Performance and Scalability in Microservices-Based Systems.* Bandung, Indonesia, 2022 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS).