Quorum Business Solutions expands energy management solutions using Azure Service Fabric

April 30, 2018

This post was authored by Shawn Cutter from Quorum, in conjunction with Michael Thomassy, Ed Price, and Matthew S

This post is part of a series on customers who've participated in the Azure Service Fabric (https://azure.microsoft.com/en-us/services/service-fabric/) preview program over the last year. We look at why they chose Service Fabric, and take a closer look at the design of their application, particularly from a microservices perspective. You can read all the profiles in this series here (https://blogs.msdn.microsoft.com/azureservicefabric/tag/customer-profile/) .

In this post, we profile Quorum Business Solutions, their well-management operations running on Azure, and how they designed their architecture using Service Fabric.

Quorum Business Solutions

Quorum Business Solutions makes innovative software for hydrocarbon and energy business management. Quorum has nearly complete coverage of the energy landscape with both breadth and depth for the upstream and midstream sectors of the oil and gas industry. In 2015, Quorum acquired Fielding Systems, an early adopter of Azure Cloud Services (https://azure.microsoft.com/services/cloud-services/) , that focused on field data capture, production operations, and SCADA/M2M services running entirely from the cloud.

In 2010, Microsoft Azure consisted of PaaS Compute, Storage, and a few other services. In order to maximize utilization of compute instances and limit hosting overhead, built a basic service deployment and hosting framework that is still used by several Quorum services today. It worked through configuration files and zip package deployments to manage each compute instance, allowing for multiple services to be deployed across multiple Cloud Services.

As a bootstrapped startup, Fielding Systems experienced great success on top of Microsoft Azure, growing customers and wells monitored between 2-4x every year, ultimately leading to the acquisition by Quorum.

Drive for increased scale, manageability and resiliency of services fueled by growth

Quorum has continued the growth of its field-operations platform, managing tens of thousands of wells across North America. This growth in terms of customer demand and the development team headcount drives the need for continuous improvements by decoupling services and processes. The desired result is more consistent scale as they bring on larger customers with more assets.

Quorum's operations include tens of thousands of wells under management, highlighting the scale that is possible with Service Fabric.

Advantages of Service Fabric

The following advantages informed Quorum's decision to build a microservice-based application using Service Fabric:

- **Stateful services** provide better performance and resiliency associated to Azure SQL Database (https://azure.microsoft.com/en-us/services/sql-database/) bottlenecks.
- **Service deployment model and process management** improved the deployment of applications and services over a custom-built infrastructure.
- **Planned support for on-premises deployments via Azure Stack** enables Quorum to easily deliver the same functionality to customers, whether they choose run in the public or private cloud.
- **Local development experience** offers the advantage of running on actual binaries and not on some form of emulator. This high fidelity local testing improves agility for developers.
- **Scalability** from the granular scaling control for each service, along with an easy path for future scaling with continuous improvements.
- **Agility** from the ability to react to changes in usage and custom demand on the application by further partitioning data. Services can be automatically moved and managed much more rapidly within a Service Fabric cluster.
- **Reduction in latency** as more and more services are converted to stateful services; performance will continue to increase by eliminating trips to external cache stores and services.

Designing the applications and services

Quorum considered the following factors in building out their design:

- **Greatest need**: The specific components and services moved to Service Fabric were selected based on need and long-term benefit through reduction in developer and process-control oversight.
- **Migration with limited disruption**: It is necessary to keep the 24/7 operation and monitoring service running at all times. Service failover, monitored upgrades with automatic rollback in case of issues, help ensure service availability and reliability.
- **Logging and telemetry**: It is fundamental to any cloud application to log errors and to know the performance and health of any service at any time.
- **Hot and Cold Path Logging**: Most Service Fabric applications utilize hot and cold paths for application log telemetry data that also include a unique transactionID in the context of all. The hot path is designed for critical application milestones.

- **Applications grouped by function in a logical solution**: Quorum's NuGet packages were used to provide consistency, without interfering compromising backwards compatibility with an existing Cloud Service.

" Service Fabric allowed us to leapfrog over our original cloud platform in a way that ultimately simplifies the architecture by eliminating the overhead of handling cache, state, and compute independently."

—Shawn Cutter: Vice President
Quorum Business Solutions

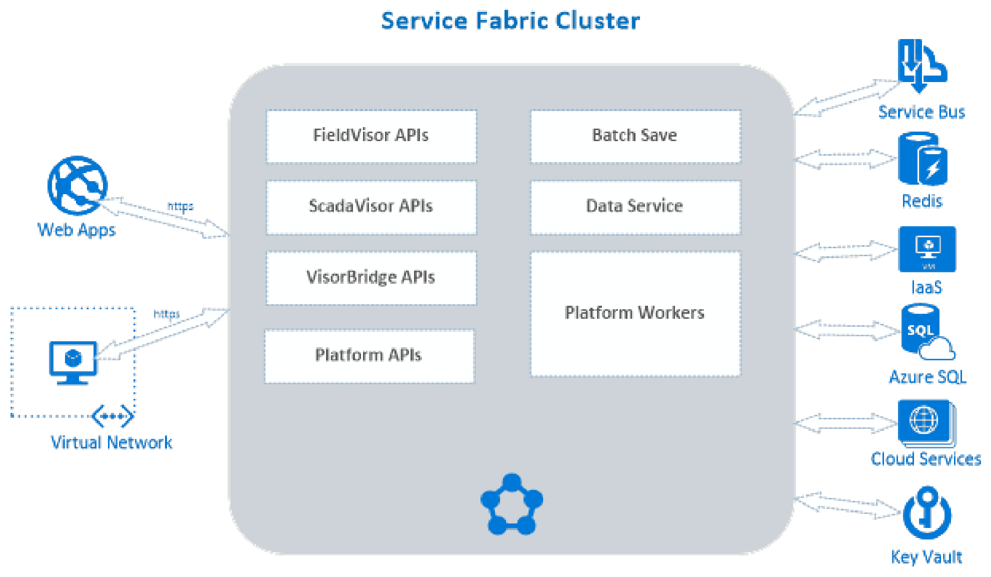

Figure 1: How the other Azure services interact with a Service Fabric cluster

Azure services in this solution

- **Service Fabric Cluster** contains all the API services, batch processing, and worker jobs for the entire platform.
- **Service Bus** (https://azure.microsoft.com/en-us/services/service-bus/) provides backup persistence for batch data write queues in the event that an extended database outage occurs. While this has not happened the scenario was thoroughly tested to ensure that in the worst conditions, the service would withstand the outage. Service Bus is also heavily leveraged to provide support for distributing a polling workload across multiple polling instances that span multiple virtual networks.

- **Key Vault** (https://azure.microsoft.com/en-us/services/key-vault/) is used for securing database connection strings, via certificate storage for service endpoints.
- **Redis** (https://azure.microsoft.com/en-us/services/cache/) is being replaced, but it is still used outside of the Azure Service Fabric cluster on Azure Cloud Services.
- **SQL DB** is used by all the customer-facing databases.
- **Storage** (https://azure.microsoft.com/en-us/services/storage/) is utilized by many legacy services and core platform processes, such as report generation, data storage, and file storage.
- **IaaS** VMs are used by the VisorBridge polling instances that contain the device drivers, communication, and data processing of all the device and sensor data.
- **Cloud Services** is used by legacy ASP.NET web applications.
- **VNet/Site-to-Site** connections are used to provide secure communications from the customer device network to the Azure VM instances that are responsible for monitoring the field devices and sensors.

Service Fabric Implementations

Quorum required a batch actor saving scenario; they leveraged Service Fabric and wrote the following services to meet that need.

- **Stateful actor service**: Provides the availability and resilience needed to ensure that no data collected from field devices will get lost in the event of a polling machine failure, while also providing improved throughput to each customer’s backend Azure SQL Databases.
- **Stateless gateway service**: Provides two listener endpoints. The first endpoint is based on Open Web Interface for .NET (OWIN), hosting a WebAPI controller that exposes the ability to receive data save requests. The second is a web-socket listener that was added to improve the data ingestion throughput. This API is used by the VisorBridge polling service to send all database saves to SQL Database for all the database save transactions that are executed in batch by stateful actor services.
- **Stateful batch actor services**: Provides reliable storage of data collected from field devices for all customers. Separate actors are utilized for each core entity inside of ScadaVisor, using a base actor type that is inherited by each actor. The base actor type contains the Reminder events that check periodically to push data batches to Azure SQL Databases using table-valued parameters for maximum through. Each customer has their own actor addressed by their same ID.
- **Stateful batch actor aggregator service**: Monitors each actor’s performance and statistics across all actors and stores the data at a customer level. These actors also use reminders to periodic updates.

Steps for polling device data and saving to a customer database

The following describes the general flow through Quorum's application:

1. A polling transaction is initiated from the master scheduler service that is in charge of all routine device data requests. This accounts for 99% of the of the transaction load; authorized users can also submit real-time requests from the web or mobile applications.
2. The transaction request is routed through Azure Service Bus to the assigned polling machine instance.
3. The polling machine receives a transaction, makes appropriate calls to the data service to pull in current configuration data that is specific to the type of device and request, and finally it queues this work item for processing.

- The VisorBridge data service is the repository service that sits on top of the configuration and transactional data required by the device communication service, which is a highly scalable, multi-threaded service whereby thousands of devices can be polled simultaneously. As work requests are received from Azure Service Bus asynchronously, these work items are set up using configuration data and are queued for processing. The legacy version of the data service was implemented to provide better scale by leveraging Redis as the caching layer on top of Azure SQL Database.
- The polling thread processes the request by communicating to the device, and it raises a MetricsReceivedEvent with the collected data payload.
- A local event listener converts and posts the datathe stateless gateway service using a web socket client. Local event listeners are added as plugins dynamically at runtime.
- The stateless gateway service uses the ActorProxy to resolve the Batch Storage Actor that corresponds to the metric type was collected. Metric types correspond to specific tables in SQL Azure Database.
- An additional set of data processing actors that correspond to each metric type use reminders to periodically check for new data stored in their storage actor counter parts.
- Any database operation that fails the typical transient fault-handling conditions is passed off to a problem queue monitored by another service. Problem data batches are interrogated more thoroughly to identify the rows in the batch that are causing the batch to fail. Examples of these types of failures include:
- Bad dates logged by devices.

- Bad sensor data on remote devices producing values outside of normal limits.

" Service Fabric Actors provide a natural approach to refactoring legacy, resource intensive, database processes to stateful services, enabling our developers to easily pivot from batch to event streams."

—Brian Shinn: DevOps
Quorum Business Solutions

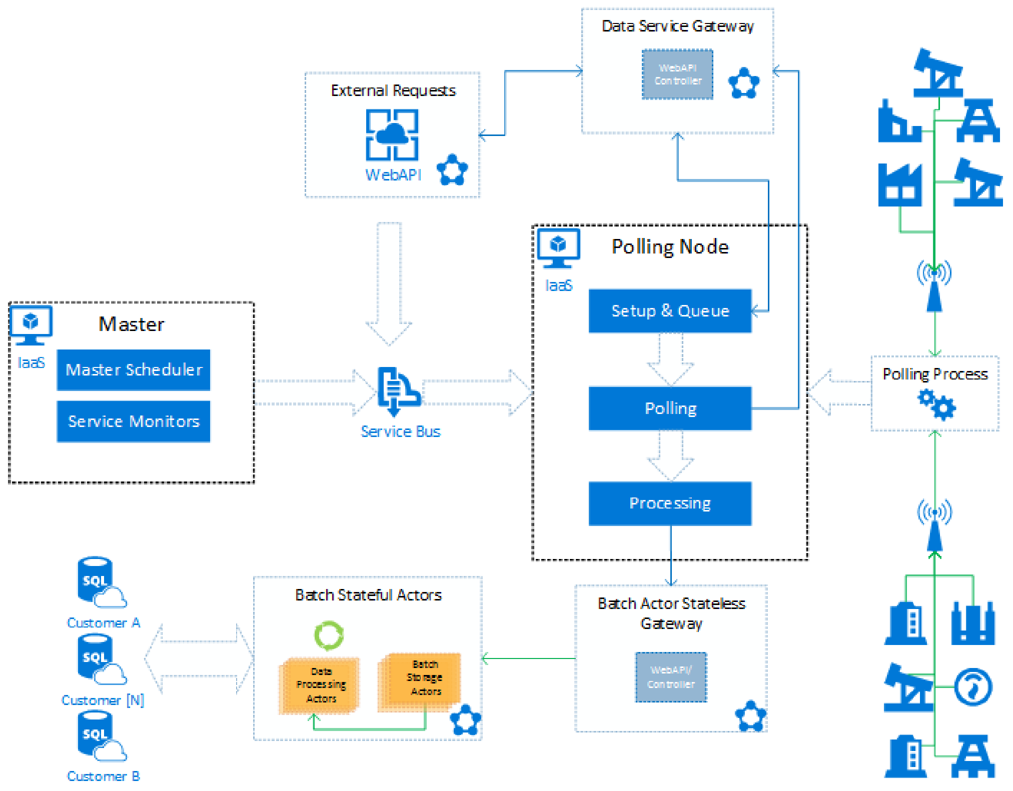

Figure 2: How the actors persist device data

Worker Process Migration Details

The custom Cloud Services worker role is another example of how Fielding Systems built innovative solutions on Azure in 2010-2011. The worker process runs in the background on any Compute service. One master service exists that is in charge of maintaining the schedules for all the application and platform processes. The master process used Azure Storage Queues to distribute workload to each processor added when the cloud service first initializes. Since these are internal services, no external gateways were needed directly to the new worker services. A stateless gateway is used to provide direct access to the stats collected from the running services. The following services were used for migration:

- **Main scheduler stateful actor service**: Replaced the master scheduler singleton thread in the work role. Execution schedules are controlled with service configuration settings. A single stateful actor uses a reminder to continually check for more work to distribute to the other stateful services.
- **Notification stateful service**: This service distributes all forms of communication to internal and external users. The service was upgraded to use a Service Fabric ReliableQueue datastructure to ensure that messages are processed and can be called from others services, such as the report execution and the ScadaVisor a
- **SQL retry stateful service**: The batch actor save eliminated the need for this for ScadaVisor, but it is still being used by a few legacy services and FieldVisor. All bulk database operations use TVP's for batching throughput, and any failures will get archived to a SQL Database table. This service periodically polls a database for any retry operations and attempts to reissue those commands.
- **Report execution stateful service**: This processes all scheduled reports across all applications for all customers. The requests are queued and processed against each primary data source, which is logically partitioned by the customer and application. All reports are running on top of SQL Server Reporting Services (SSRS), which is running on an IaaS SQL Server VM. The use of a ReliableQueue and ReliableDictionary enable maximum throughput and the resilience from any widespread outage.
- **ScadaVisor alarm engine stateful actor service**: This analyzes the data collected from devices and sensors for alarm conditions and sends an appropriate notification to customers via email, SMS, or voice callouts. When triggered for processing, new data is pulled from each customer's SQL Database and analyzed for alarm conditions. All alarm configurations and settings are maintained in the Actor's state, enabling the ability for data analysis to take place on data streams.
- **FieldVisor task processor stateful actor service**: This service executes and manages several backend processes that are configured inside of each customer's Azure SQL Database. Each stateful actor maintains the current configuration for each customer and uses a reminder to continually check for new work items to process.
- **ScadaVisor task processor stateful actor service**: This service operates in a similar fashion to the above FieldVisor task processor.

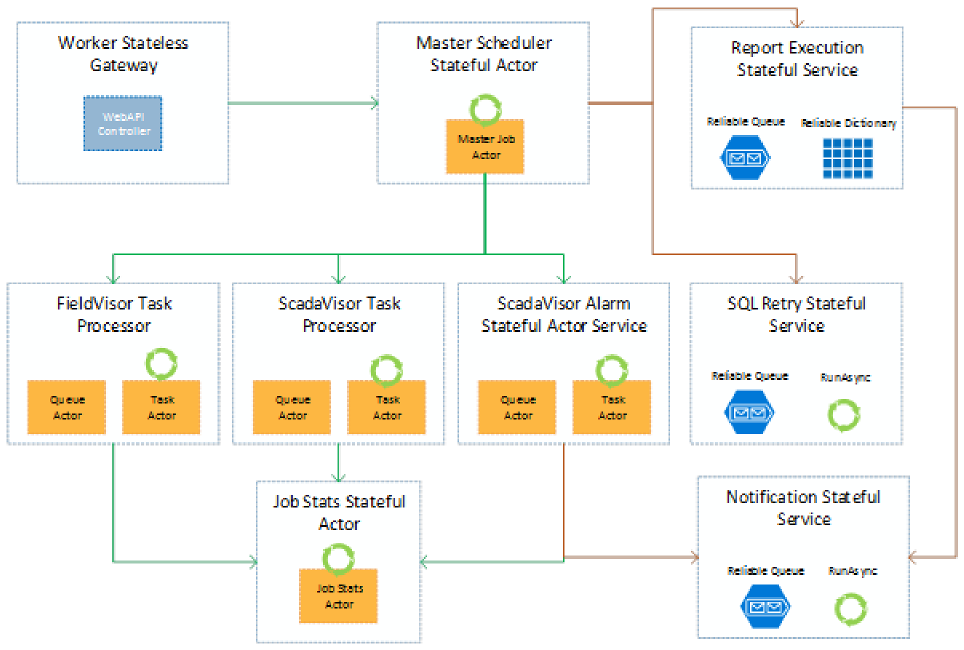

Figure 3: How the services were architected in the migration process

Summary

Many organizations built on the cloud have discovered that there is a significant amount of 'learning by doing.' As one of the early adopters of Azure, Quorum (as Fielding Systems) gradually built the knowledge of how to design applications and services to provide the scale and resilience desired. While the components built over that time are still functional, they require a significant amount of tribal knowledge to manage. By adopting Service Fabric, Quorum was able replace many of those components and create a simpler design that will scale well into the future.