



ScaleOut Software

How to Track a Million Data Sources

IMCS Training Session: Technology Overview

October 27, 2020

Dr. William Bain, Founder & CEO,

wbain@scaleoutsoftware.com

Welcome!

Documentation is posted at:

<https://www.scaleoutsoftware.com/scaleout-support/documentation>

The slides and demos are posted at:

<https://github.com/scaleoutsoftware/DigitalTwinDemos>

- **Session 1: Technology Overview** (Bill Bain, 1 hour)
 - What are Real-Time Digital Twins?
 - A Tour of the ScaleOut Digital Twin Streaming Service™
 - ScaleOut Digital Twin Builder™ Software Toolkit
 - Demo
- **Session 2:** Walkthrough: Building a Java Digital Twin for Security Tracking (Brandon Ripley, 45 min)
- Break (10 minutes)
- **Session 3:** Walkthrough: Building a C# Digital Twin for a Wind Turbine (Oleg Shmytov, 45 min)
- **Demo:** A Contact Tracing Application (Olivier Tritschler, 15 min)
- **Session 4:** Build Your First Real-Time Digital Twin (Team, 45 min)
- **Wrap-Up** (Bill Bain, 5 min)

- **Session 1:** Technology Overview (Bill Bain, 1 hour)
 - What are Real-Time Digital Twins?
 - A Tour of the ScaleOut Digital Twin Streaming Service™
 - ScaleOut Digital Twin Builder™ Software Toolkit
 - Demo
- **Session 2: Walkthrough: Building a Java Digital Twin for Security Tracking** (Brandon Ripley, 45 min)
- Break (10 minutes)
- **Session 3:** Walkthrough: Building a C# Digital Twin for a Wind Turbine (Oleg Shmytov, 45 min)
- **Demo:** A Contact Tracing Application (Olivier Tritschler, 15 min)
- **Session 4:** Build Your First Real-Time Digital Twin (Team, 45 min)
- **Wrap-Up** (Bill Bain, 5 min)

- **Session 1:** Technology Overview (Bill Bain, 1 hour)
 - What are Real-Time Digital Twins?
 - A Tour of the ScaleOut Digital Twin Streaming Service™
 - ScaleOut Digital Twin Builder™ Software Toolkit
 - Demo
- **Session 2:** Walkthrough: Building a Java Digital Twin for Security Tracking (Brandon Ripley, 45 min)
- Break (10 minutes)
- **Session 3:** Walkthrough: Building a C# Digital Twin for a Wind Turbine (Oleg Shmytov, 45 min)
- **Demo:** A Contact Tracing Application (Olivier Tritschler, 15 min)
- **Session 4:** Build Your First Real-Time Digital Twin (Team, 45 min)
- **Wrap-Up** (Bill Bain, 5 min)

- **Session 1:** Technology Overview (Bill Bain, 1 hour)
 - What are Real-Time Digital Twins?
 - A Tour of the ScaleOut Digital Twin Streaming Service™
 - ScaleOut Digital Twin Builder™ Software Toolkit
 - Demo
- **Session 2:** Walkthrough: Building a Java Digital Twin for Security Tracking (Brandon Ripley, 45 min)
- Break (10 minutes)
- **Session 3:** Walkthrough: Building a C# Digital Twin for a Wind Turbine (Oleg Shmytov, 45 min)
- **Demo: A Contact Tracing Application** (Olivier Tritschler, 15 min)
- **Session 4:** Build Your First Real-Time Digital Twin (Team, 45 min)
- **Wrap-Up** (Bill Bain, 5 min)

- **Session 1:** Technology Overview (Bill Bain, 1 hour)
 - What are Real-Time Digital Twins?
 - A Tour of the ScaleOut Digital Twin Streaming Service™
 - ScaleOut Digital Twin Builder™ Software Toolkit
 - Demo
- **Session 2:** Walkthrough: Building a Java Digital Twin for Security Tracking (Brandon Ripley, 45 min)
- Break (10 minutes)
- **Session 3:** Walkthrough: Building a C# Digital Twin for a Wind Turbine (Oleg Shmytov, 45 min)
- **Demo:** A Contact Tracing Application (Olivier Tritschler, 15 min)
- **Session 4: Build Your First Real-Time Digital Twin** (Team, 45 min)
- **Wrap-Up** (Bill Bain, 5 min)

- Goals and challenges for stream processing with large numbers of data sources
- What are real-time digital twins? Why use them?
- Advantages in comparison to traditional approaches
- Target use cases
- Using in-memory computing to host digital twins
- Implementing and deploying digital twin models
- Using ScaleOut's cloud-based streaming service
- APIs designed for building digital twins
- Demo of a security application for a power grid

About ScaleOut Software

- Develops and markets software for **in-memory computing**:
 - Scales application performance and
 - Provides real-time analytical insights using
 - In-memory data storage, computing, and stream processing
- Deep domain expertise:
 - Dr. William Bain, Founder & CEO. Bell Labs, Intel, Microsoft
 - Over 15 years in the market
 - Consistent track record of innovation and technology leadership
- Flexible business model to meet diverse needs:
 - Fully supported software releases
 - Dedicated to ease-of-use to minimize training and lower TCO
 - On-premise or cloud
 - Choice of licensing models: perpetual, subscription, cloud-hosted



Goals of Stream Processing

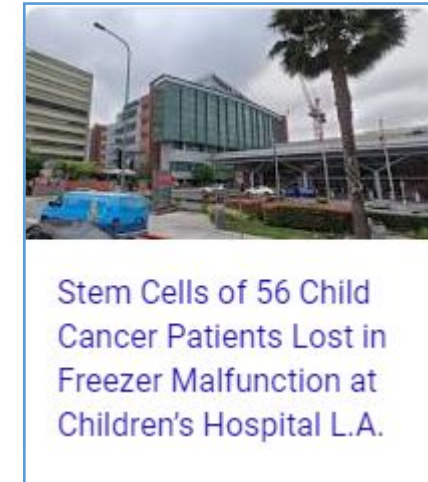
- Goal: **maximize situational awareness & real-time control**
- How:
 - Process incoming data streams from many thousands of devices.
 - Analyze events for patterns of interest.
 - Provide timely (real-time) feedback and alerts.
 - Provide aggregate analytics to identify patterns.
- Many applications in IoT and beyond:
 - Medical monitoring
 - Logistics & manufacturing
 - Disaster recovery & security
 - Financial trading & fraud detection
 - Ecommerce recommendations



Event Sources

Quick Example: Medical Refrigerators

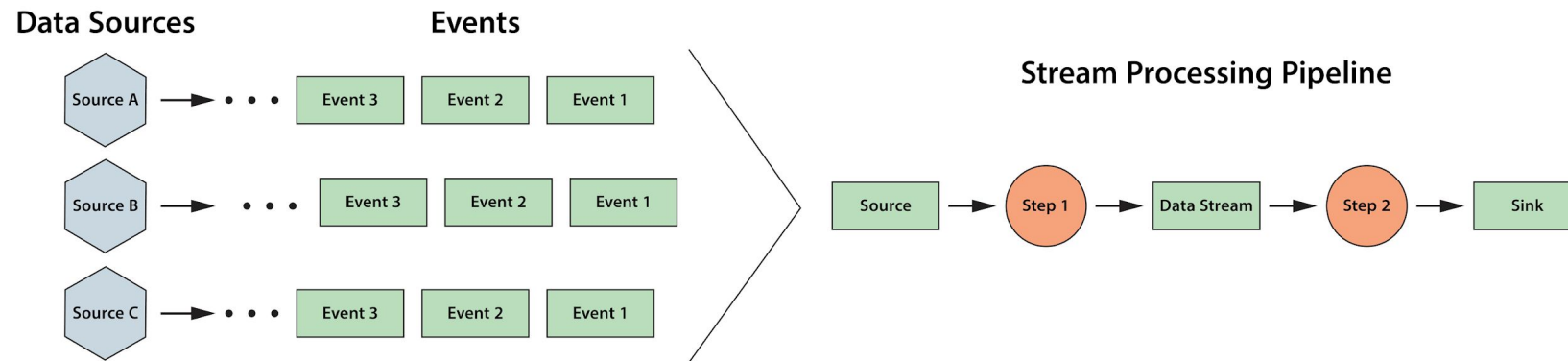
Sep 26, 2019



- Cloud-based streaming service monitors 10,000+ medical refrigerators:
- Refrigerators hold highly important tissue samples, embryos, etc.
- Service receives periodic telemetry:
 - Temperature
 - Power consumption
 - Door position, etc.
- Must predict failure before it occurs:
 - Notify user to migrate contents to another refrigerator.
 - Avoid false positives.
 - Identify widespread power outages (aggregate analytics).

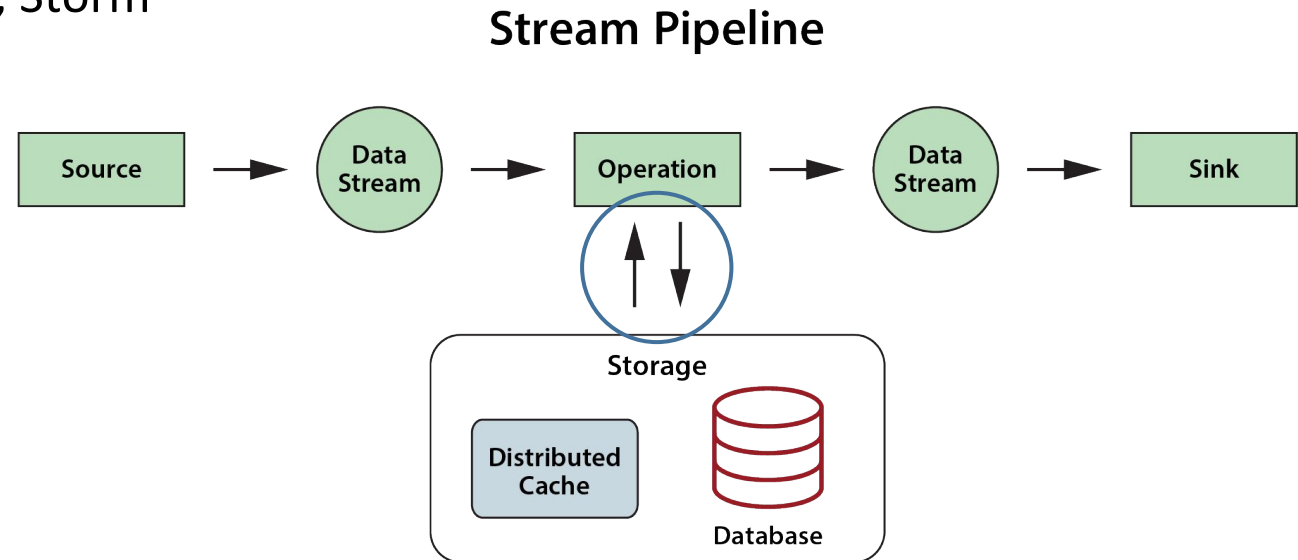


- Popular software platforms (Flink, Storm, Beam) are **pipeline-oriented**:
 - Push all messages through a single pipeline or directed graph of processing stages.



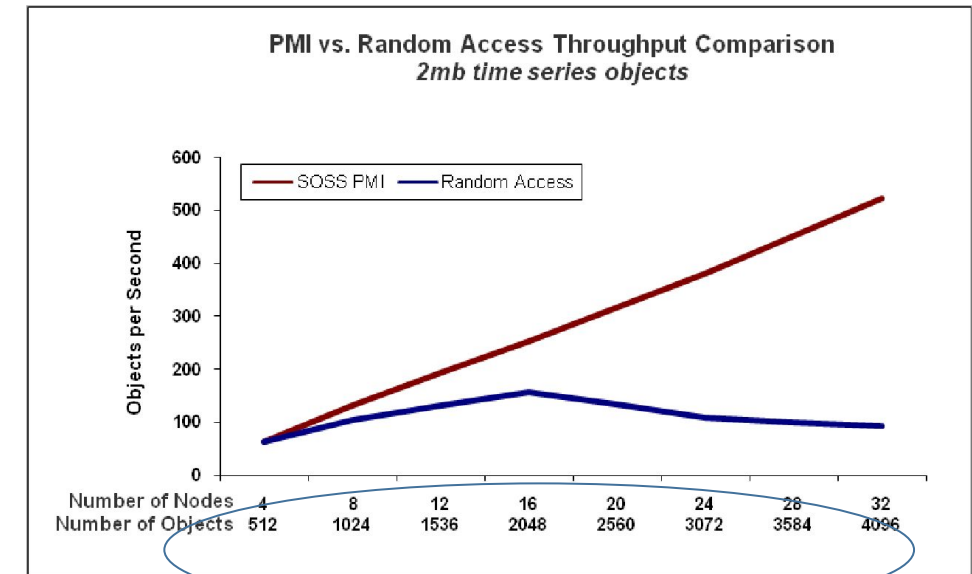
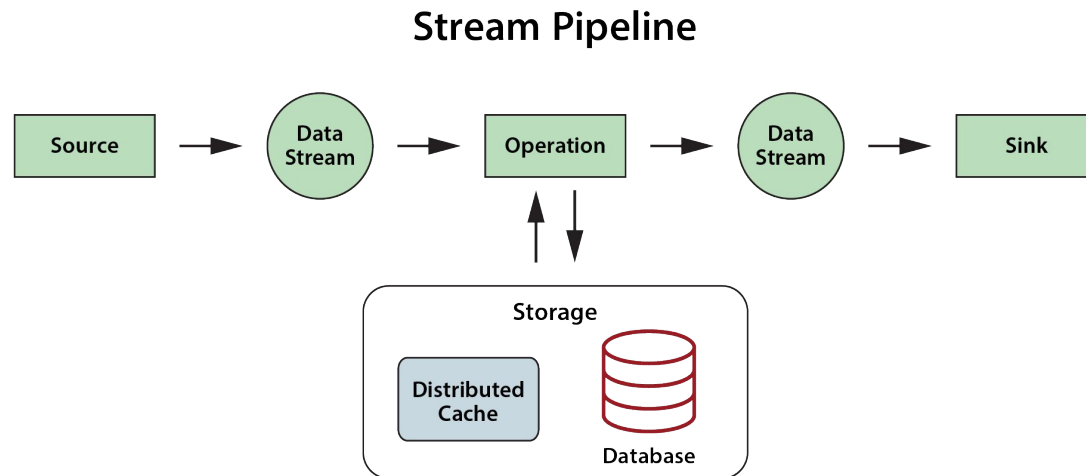
- Creates **complexity challenges**:
 - Difficult to: correlate events by each data source, track state, embed analytics
 - Cannot respond to individual data sources.
- Creates **performance challenges**:
 - Difficult to: respond with low latency, scale for thousands of data sources
- Requires **aggregate analytics** to be performed **offline**.

- Stateful stream-processing platforms add “unmanaged” data storage to the pipeline:
 - Pipeline stages perform transformations in a sequence of stages from data sources to sinks.
 - Data storage (distributed cache, database) is accessed from the pipeline by application code in an unspecified manner.
 - Examples: Apama (CEP), Apache Flink, Storm
- Problems:
 - There is no software architecture for managing state information.
 - This adds complexity to the application.
 - Creates a network bottleneck.



Avoiding Network Bottlenecks

- External data storage requires network access to obtain an event's context.
- Network bottleneck prevents scalable throughput.



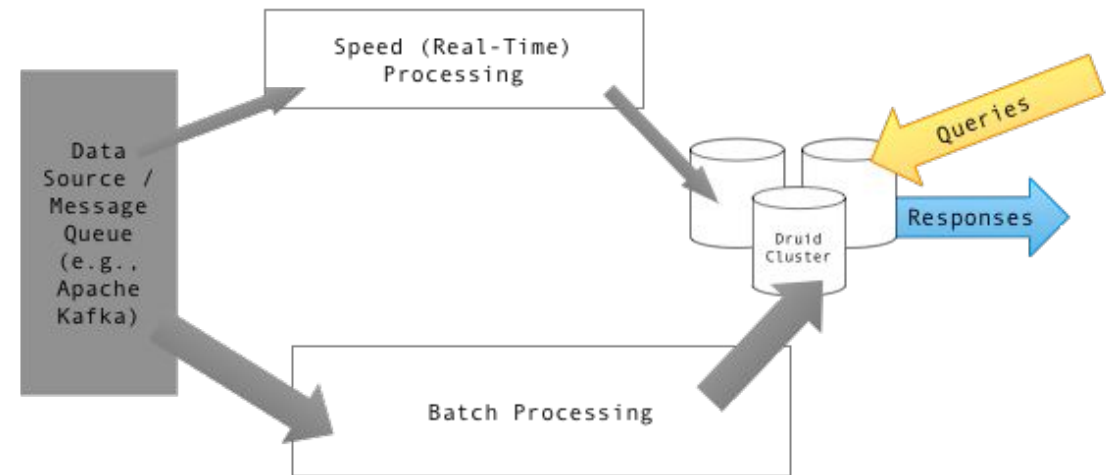
Microsoft's platform finds patterns and trends within an incoming data stream of messages.

- Aggregates data within the data stream over a time window:
 - Can trigger functions and export to a database, data lake, or Power BI for visualization.
- Examples:
 - Ad server: Count the number of ads served by product in the last 5 minutes.
 - Fleet telemetry: Compute the average speed for all trucks by region.
- Uses SQL with time windowing extensions to query data stream:
 - Each message is a row with attributes including timestamp.
 - The time-ordered stream of messages is a column to be queried.
- Azure Stream Analytics cannot analyze *why* the data sources are sending this data and give them appropriate feedback:
 - Ex.: Why users prefer certain products and what other ads should they see
 - Ex.: Why trucks are delayed (weather, lost driver, mechanical issue) and what action should be taken



Lambda Architecture: Batch Parallel Analytics ScaleOut Software

- Lambda architecture separates stream-processing (“speed layer”) from data-parallel analytics (“batch layer”).
- Creates queryable state, but:
 - Does not enhance context for stateful stream processing.
 - Does not perform data-parallel analytics online for immediate feedback.
 - Does not lead to a “Hybrid Transactional and Analytics Processing” (HTAP) architecture.



<https://commons.wikimedia.org/w/index.php?curid=34963987>

How combine stream processing with state to simplify design, maximize performance, and enable fast data-parallel analytics?

How simultaneously track and analyze the dynamic state of 1000s of data sources?

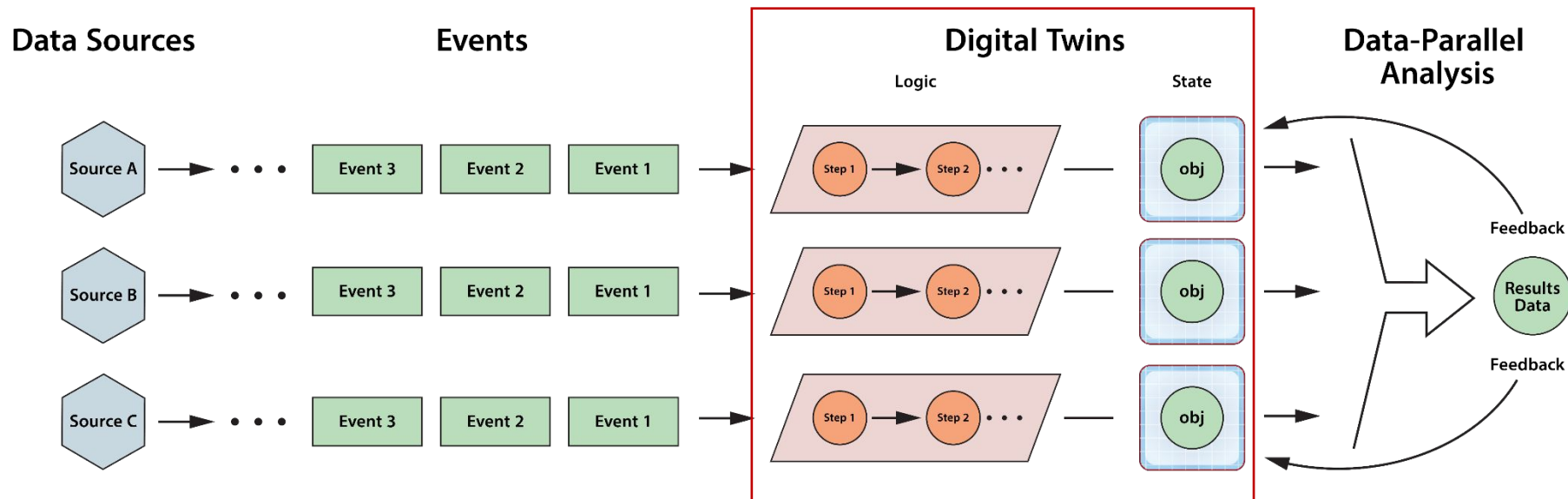
- Typical work-around: build ad hoc network of cloud-based services plus offline analytics:
 - Front-end REST service receives incoming messages and sends replies.
 - Intermediate application server farm tracks data sources and processes messages.
 - Database service and/or blob store host state.
 - Offline analytics service (e.g., Spark) analyzes stored data.
 - Offline visualization displays results.



- Problems with this approach:
 - Complex to design and implement, requiring multiple skills
 - Introduces scaling bottlenecks and availability challenges.
 - Offline analytics delay results.

Real-Time Digital Twins

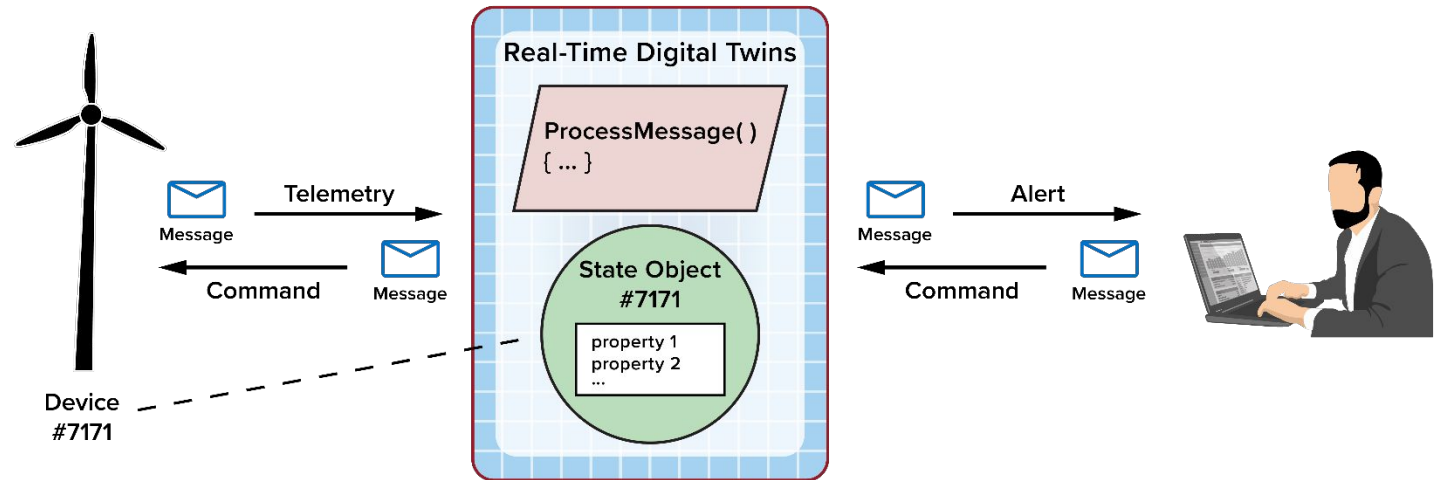
- A **new software technique** for stream-processing refactors stream-processing to center on state tracking instead of pipelined execution.
- What it does:
 - Automatically **correlates telemetry** from each device or data source.
 - **Tracks dynamic state** for each data source.
 - Provides a **software framework** for hosting application logic (e.g., rules, ML).
 - Enables **real-time aggregate analysis** in place.



Anatomy of a Real-Time Digital Twin

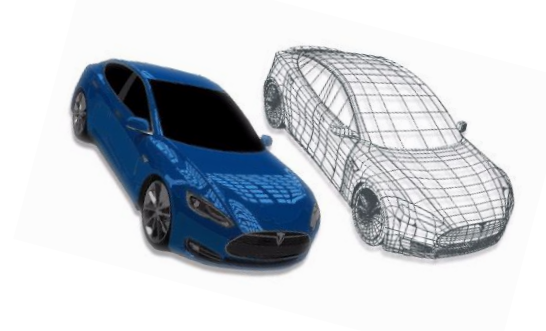
A **real-time digital twin model** describes how to process incoming key events from a specific type of data source (e.g., a wind turbine). It consists of a:

- **Message processor:**
 - Receives and analyzes events and commands.
 - Encapsulates analysis algorithm.
 - Generates alerts and outbound device messages.
- **State object**, which holds dynamic data:
 - Context/metadata for analyzing events
 - Also: time-ordered event lists, cached parameters
 - One **instance** per data source (device)



Other Uses of the Term “Digital Twin”

- Created by Michael Grieves for product design and life cycle management (PLM); popularized by Gartner:
 - A virtual version of a physical entity
 - Also, context to interpret telemetry streaming back from the field
- Also:
 - **AWS device shadow**: cloud-based repository for per-device state information with pub/sub messaging
 - **Azure IoT device twin**: JSON document that stores per-device state information (metadata, conditions)
 - **Azure digital twin**: spatial graph of spaces, devices, and people for modeling relationships in context
- These uses are *not* for **real-time stream processing**.



A digital twin may be used for simulation, as a kind of prototype to understand expected behavior, existing before there is a physical twin. It can also capture real-world behavior so that, for example, analytics and learning can be performed. ...

Definition of “Digital Twin” from the
Digital Twin Consortium

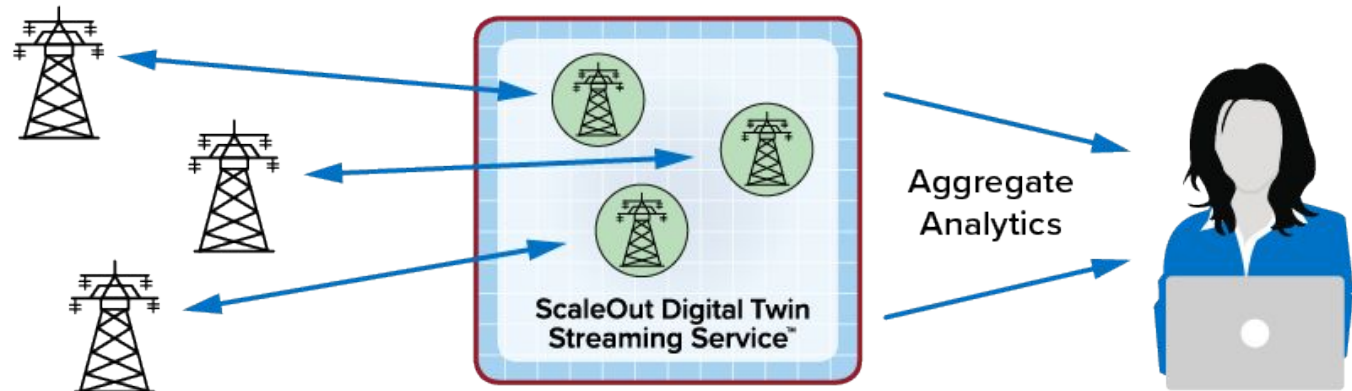
Comparison: Two Types of Digital Twins

- A real-time digital twin is **not a PLM model** of a physical device.
- However, it does model a data source's behavior for the purpose of streaming analytics.

PLM Digital Twin	Real-Time Digital Twin
Goal: Aid in product development .	Goal: Aid in real-time streaming analytics .
Models characteristics and behavior of a physical device (e.g., a simulation model).	Analyzes telemetry streams from a physical device & generates feedback and alerts.
Can proactively generate outputs over time for simulation purposes.	Reactively processes telemetry messages and commands.
Implements dynamic state that models device behavior to evaluate lifecycle issues.	Implements dynamic state that adds context to help interpret telemetry.
Example: digital twin for a medical refrigerator:	
Models door open/close events, power fluctuations, etc. to evaluate MTBF.	Analyzes events based on maintenance history, etc. to predict & avoid failure.

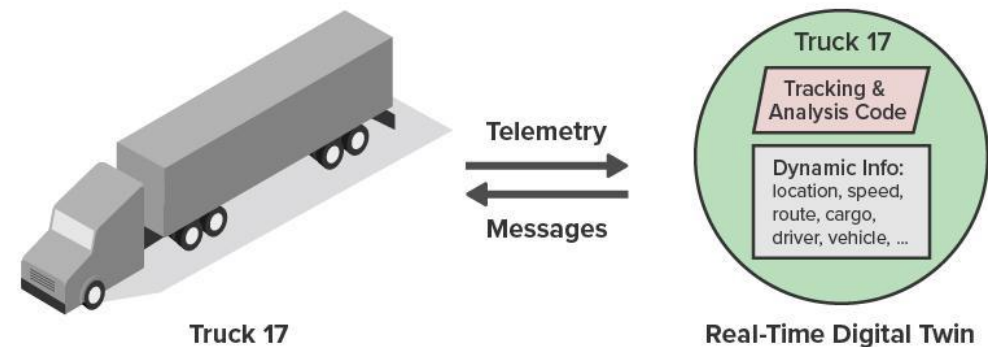
Advantages of Real-Time Digital Twins

- **Simplifies** application design:
 - Provides automatic event correlation and access to per-device state.
 - Uses well known object-oriented techniques and avoids special languages.
- Enables **deeper introspection** in real time and **enhanced situational awareness**:
 - Dynamically tracks state of each device to help analyze incoming events.
 - Provides orchestration for analytics code (e.g., rules engine, ML).
 - Enables integrated, aggregate analysis.
- High performance: respond to data source in 1-3 msec; aggregate analysis in 5-10 secs.
- Allow transparent scaling, migration to the edge, hierarchical design, and more.



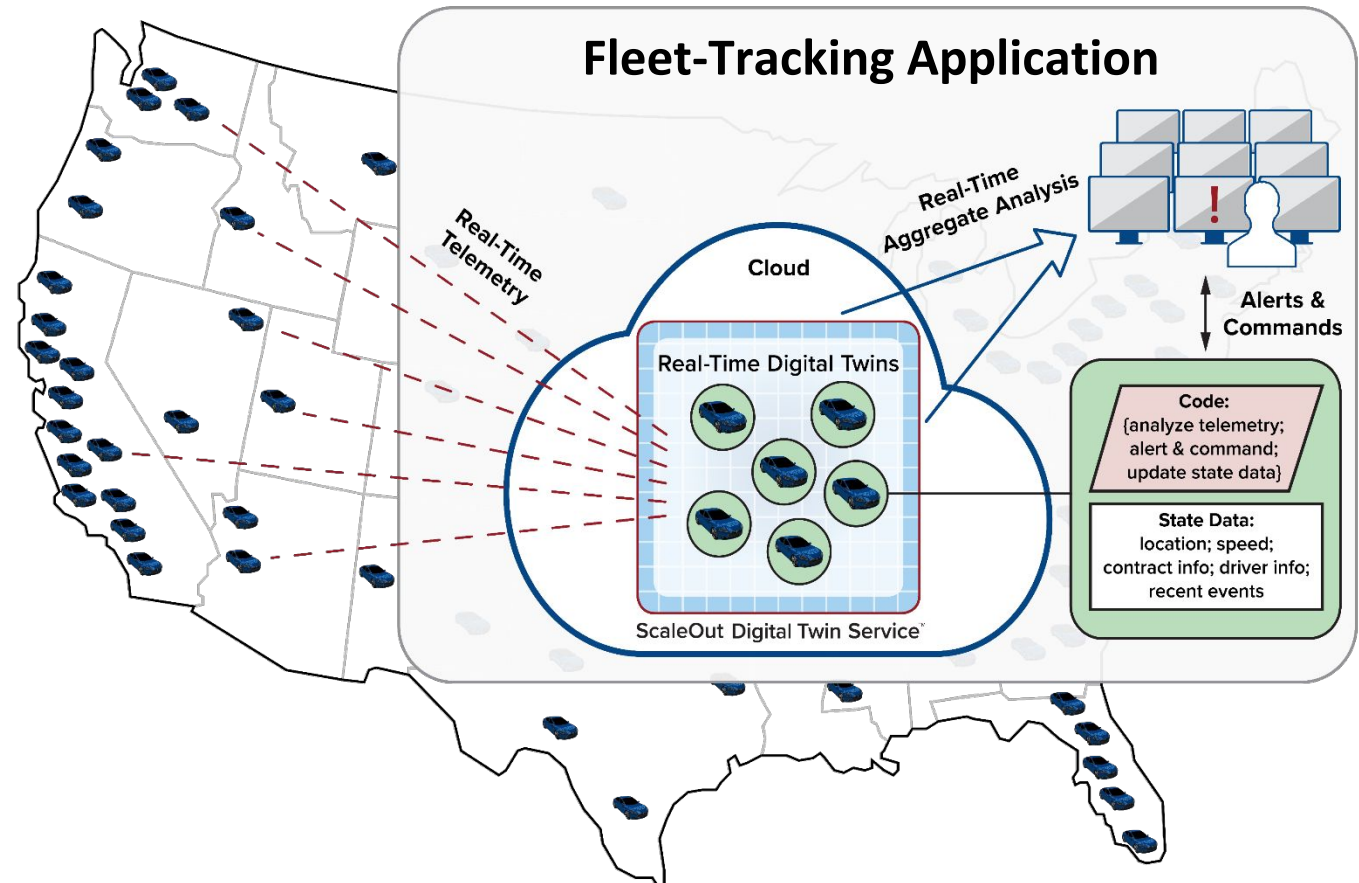
RTDTs enable focused analysis and feedback for each data source.

- How?
 - They maintain immediately accessible contextual information for each data source.
 - They allow application code to process *only* those messages from a given data source.
- Example: telematics for a trucking fleet
 - Telemetry includes location, speed, mechanical & cargo parameters.
 - RTDT includes route, cargo, info on driver, service history & issues, status
- Using this telemetry RTDT can:
 - Alert driver to upcoming hazardous road conditions.
 - Assist lost driver or alert if driving too long or unsafely.
 - Track emerging mechanical issues with vehicle or risk to cargo.
 - Maintain status which can be aggregated for all trucks to enhance situational awareness.



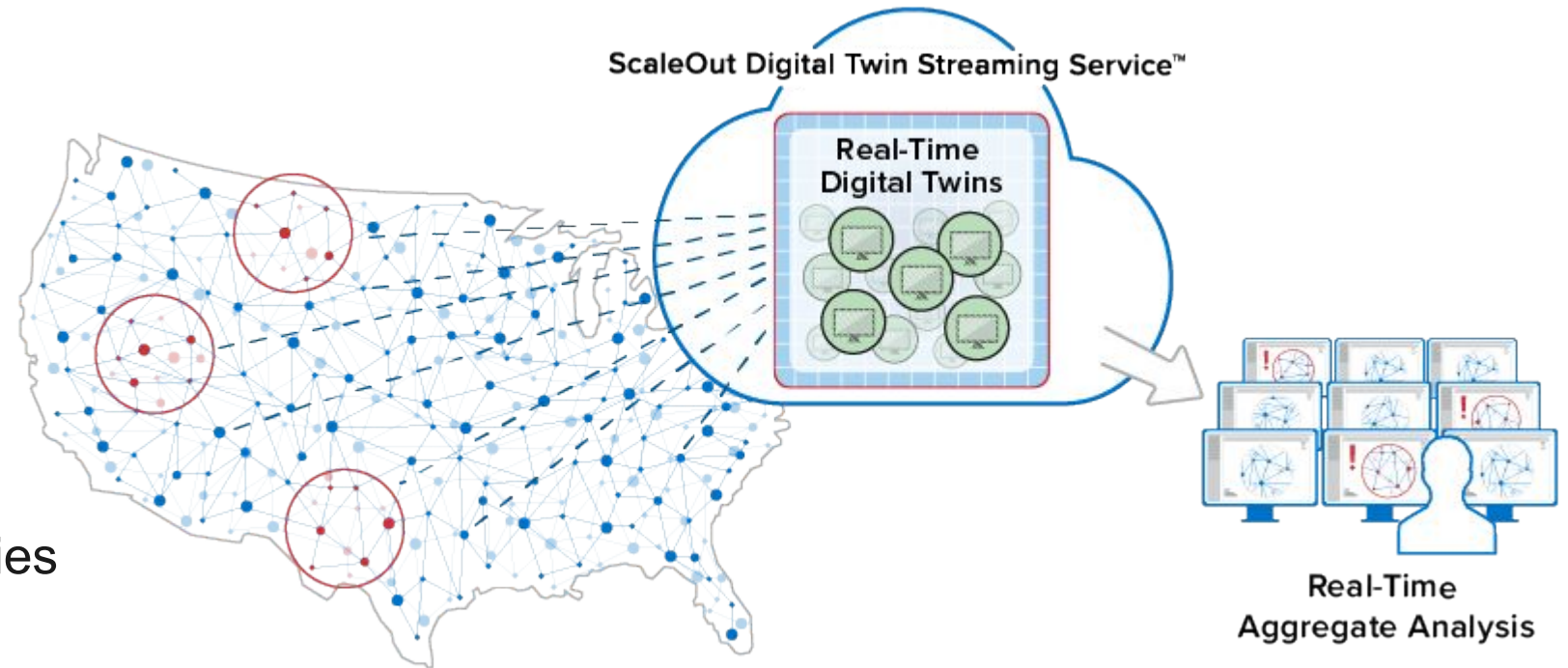
Large Scale Fleet Tracking

- Real-time tracking for a car/truck fleet
 - 100K+ vehicles
- Immediately responds to issues with individual vehicles:
 - Lost driver
 - Driver working too long.
 - Impending engine or cargo issue
- Detects & responds to regional issues within seconds
 - Weather delays, highway blockages
 - Redirects drivers.



Target Use Cases

- Applications that track **thousands of data sources which require fast response times, aggregate analysis, and situational awareness**
- General category: **real-time intelligent monitoring**
- Examples:
 - Fleet management
 - Health tracking
 - Disaster recovery
 - Security monitoring
 - Ecommerce recommendations
 - Fraud detection
 - Traffic control / smart cities



Example: Security / Disaster Recovery

- Intrusion sensors analyze telemetry to predict unauthorized access at each location.
- **Aggregate analysis** of perimeter sensors indicates scope of threat.
- Enables focused, real-time response to all critical locations.

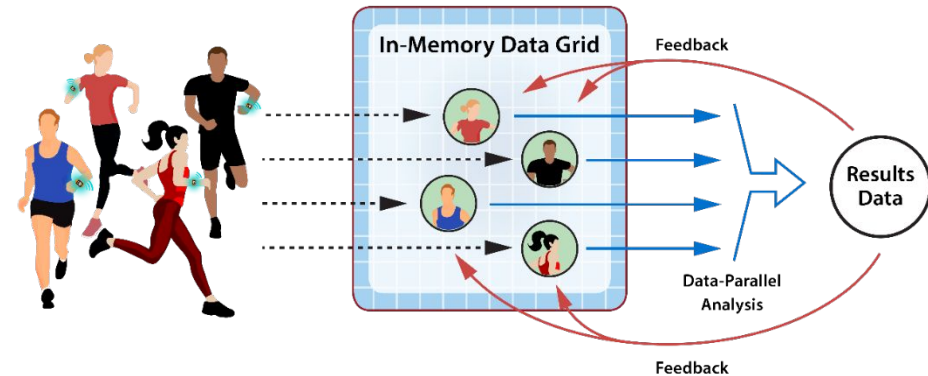
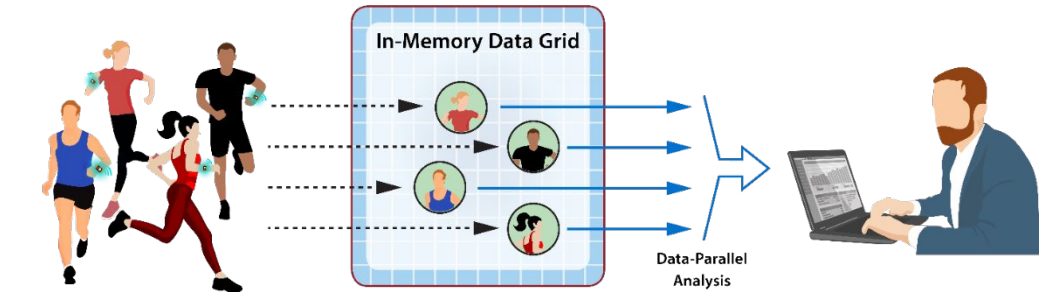
Nov 13, 2015



The November 2015 Paris attacks were a series of coordinated terrorist attacks that took place on 13 November 2015

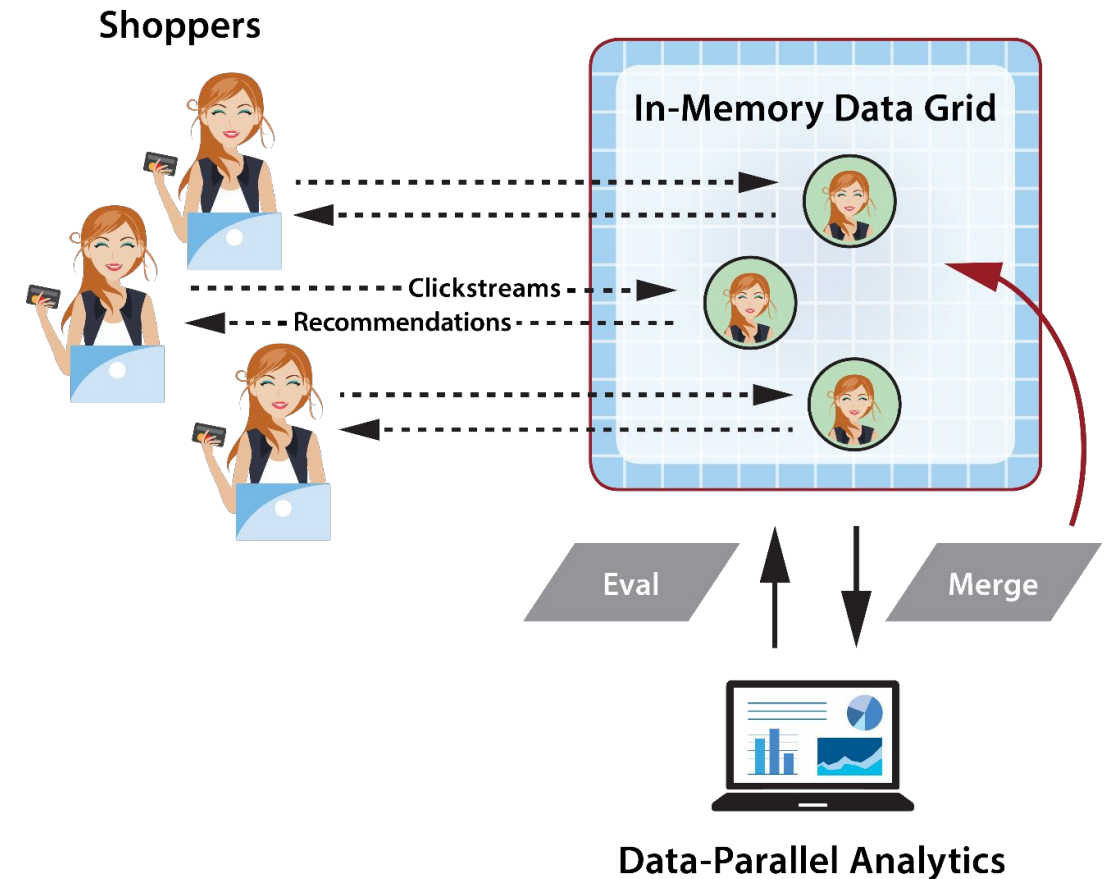
Example: Real-Time Health Tracking

- Digital twins analyze telemetry from health-tracking devices to help ensure safety (predict events):
- Digital twins receive periodic messages with key metrics (heart rate, blood oxygen, etc.).
- State objects track person's health history, medications, limitations, recent medical events.
- Analysis algorithm can integrate dynamic, aggregate results from large populations.



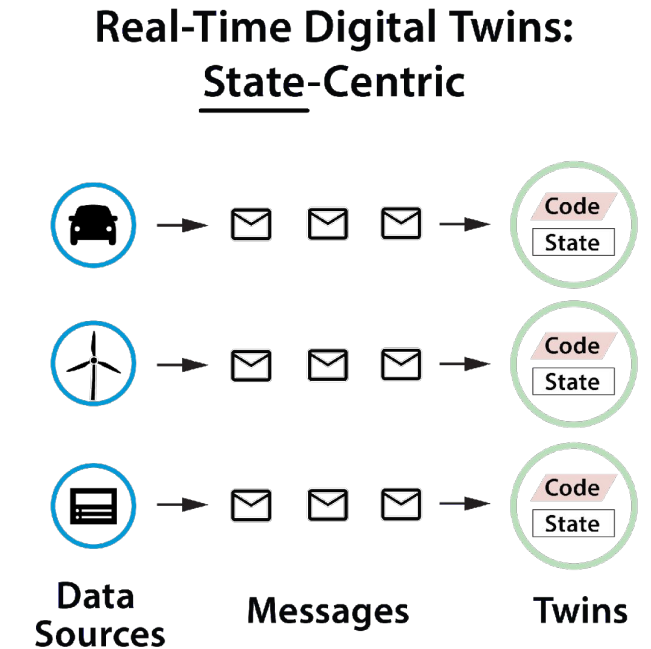
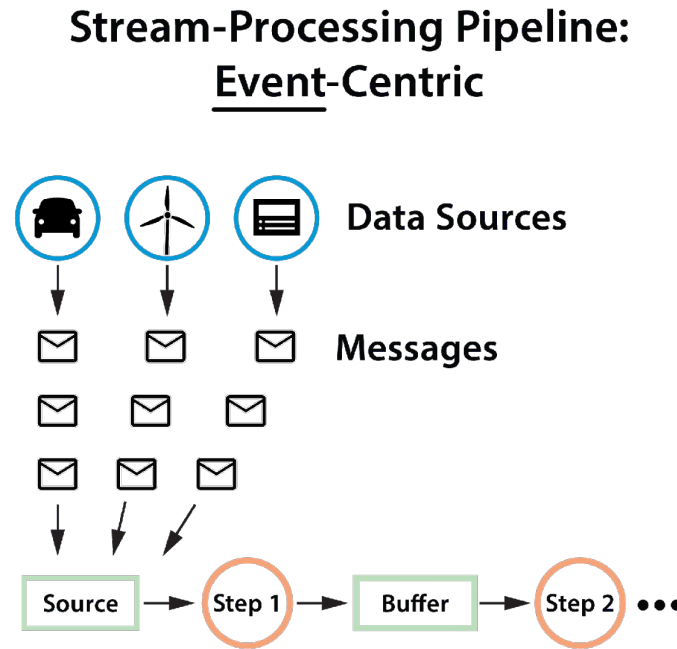
Example: Ecommerce Recommendations

- Ecommerce site may have 100k+ shoppers, each generating a clickstream.
- Digital twin for each shopper:
 - Maintains a history of clicks, shopper's preferences, and purchasing history.
 - Analyzes clicks to create new recommendations in real time.
- Aggregate analysis:
 - Determines collaborative shopping behavior, basket statistics, etc.
 - Enables targeted, real-time flash sales.



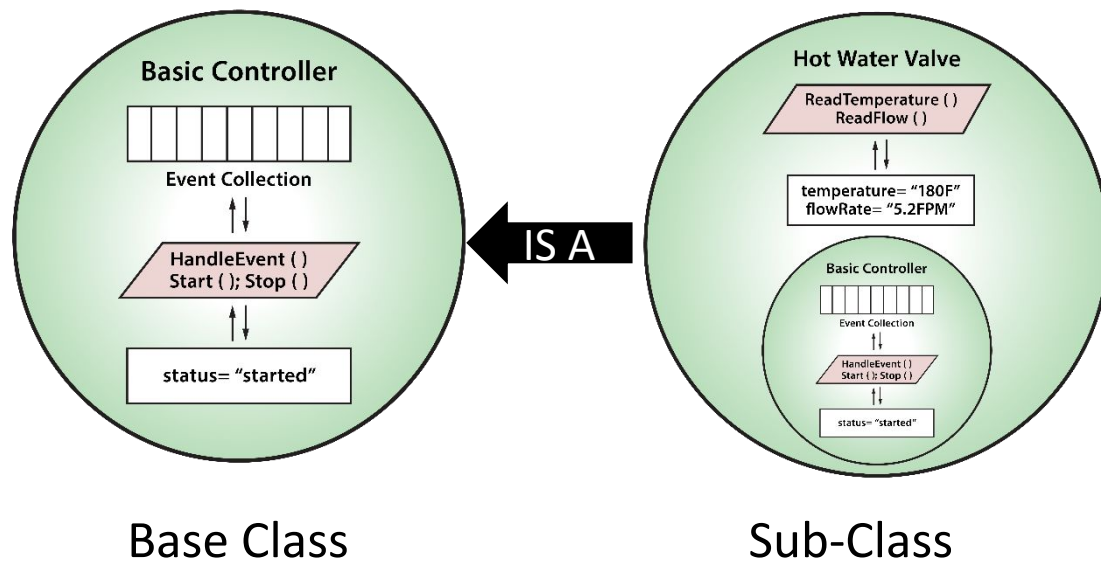
State-centric approach (vs. event-centric) simplifies tracking of thousands of individual data sources:

- Avoids event correlation in the application.
- Avoids need for *ad hoc* state storage.
- Encapsulates analysis logic in one place.
- Provides automatic domain for aggregate analysis.

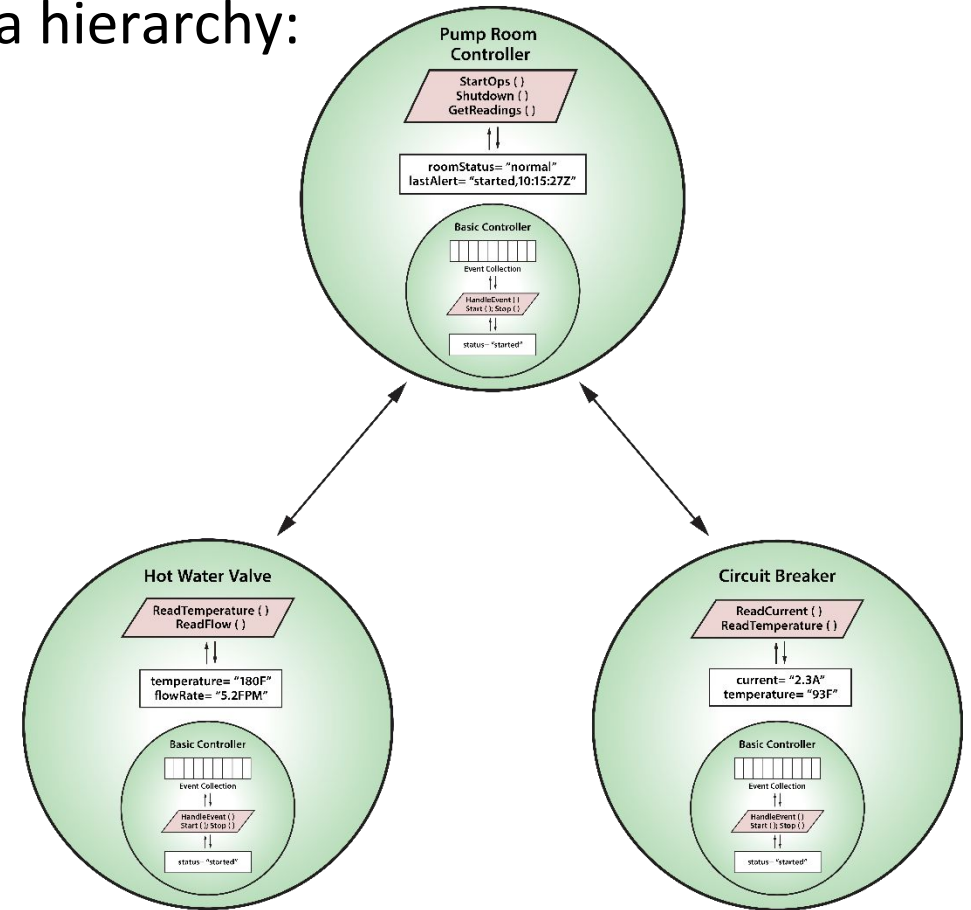


RTDTs Leverage Object-Oriented Techniques ScaleOut Software

- Digital twin objects can use inheritance to create specialized behaviors:



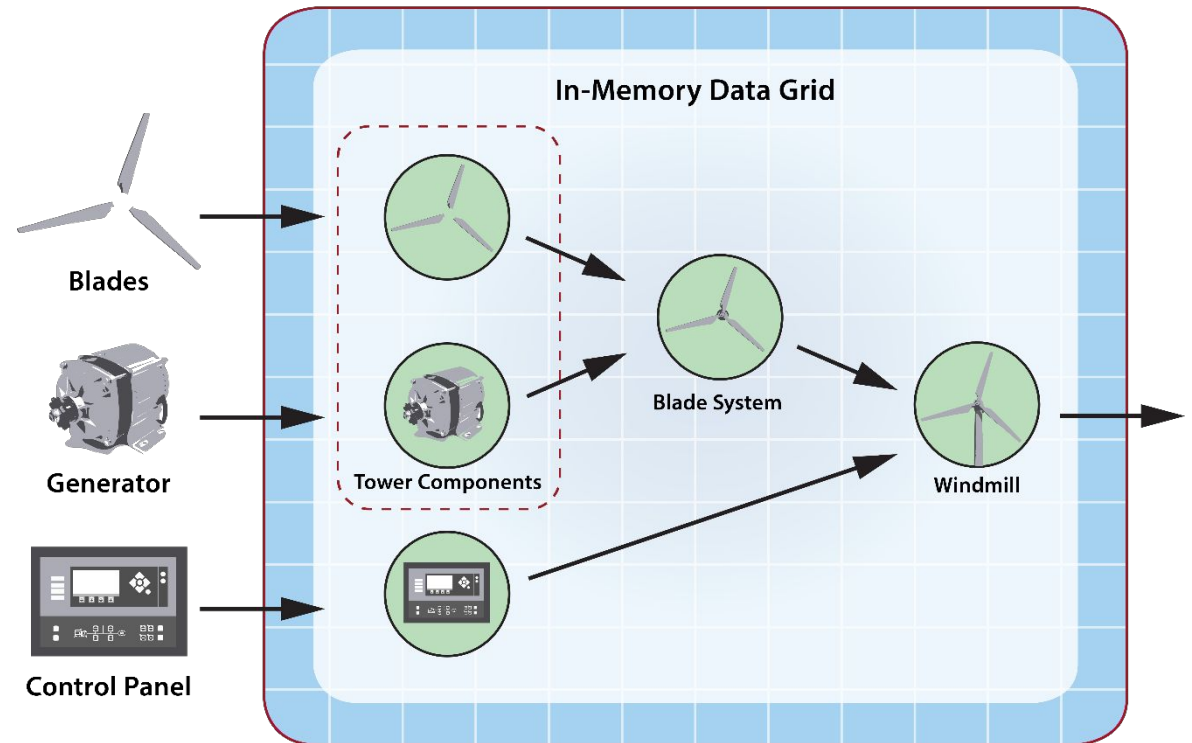
- Instances of objects can be organized in a hierarchy:



Using RTDTs in a Hierarchy

Tracks complex systems as hierarchy of digital twin objects:

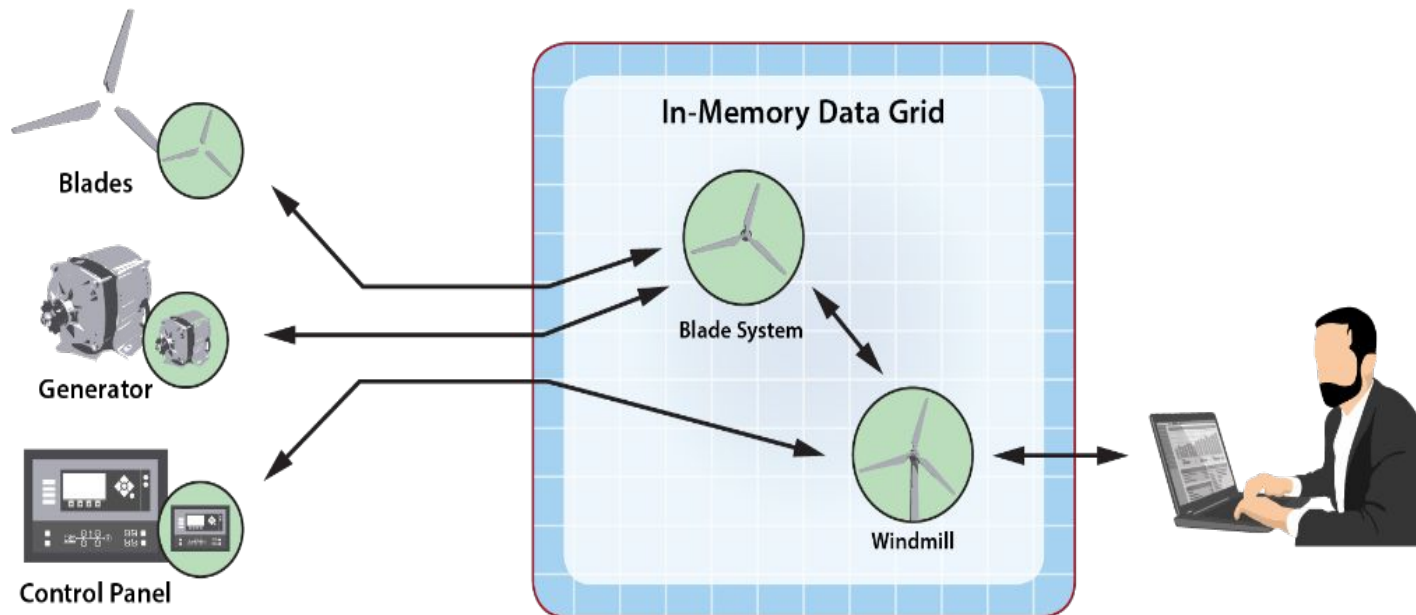
- Leaf nodes receive telemetry from physical endpoints.
- Higher level nodes represent subsystems:
 - Receive telemetry from lower-level nodes.
 - Supply telemetry to higher-level nodes as alerts.
 - Allow successive refinement of real-time telemetry into higher-level abstractions.



Example: Hierarchy of Digital Twins for a Windmill

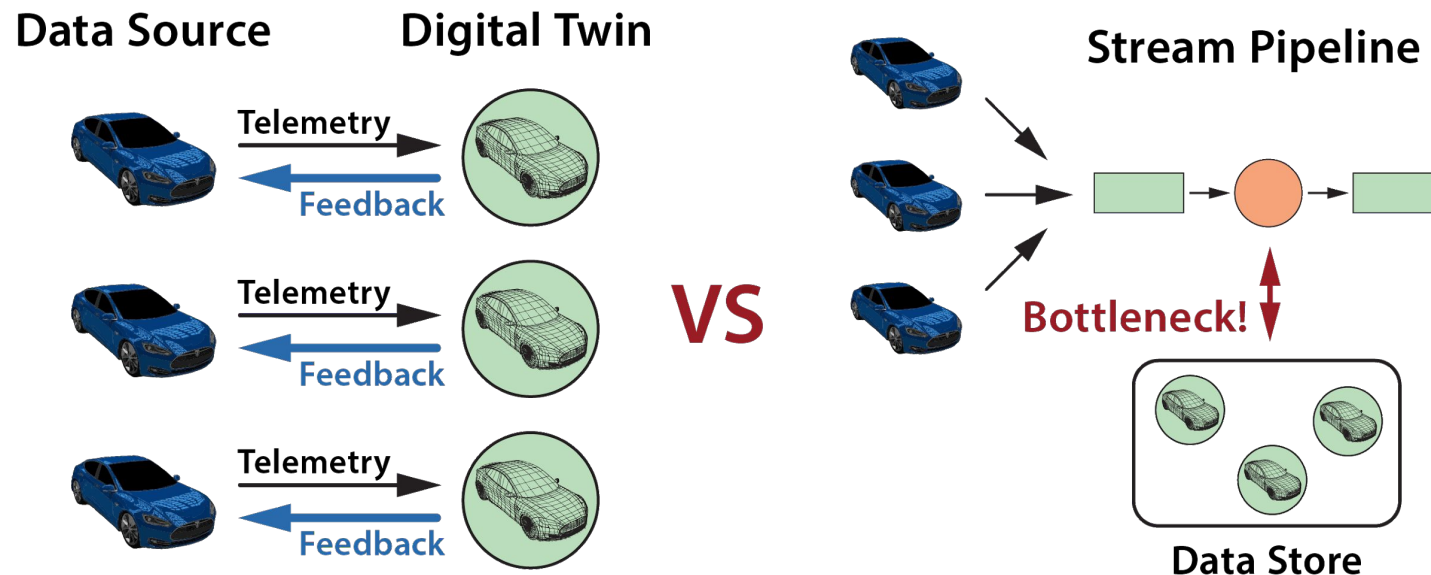
RTDTs Simplify Migration to Edge

- Migration of stream-processing intelligence to the edge is an ongoing trend driven by continuous advances in technology.
- Constructing software components as o-o digital twins simplifies migration:
 - Makes software decomposition independent of execution location.
 - Avoids rewriting code for execution at the edge; can leverage containers.



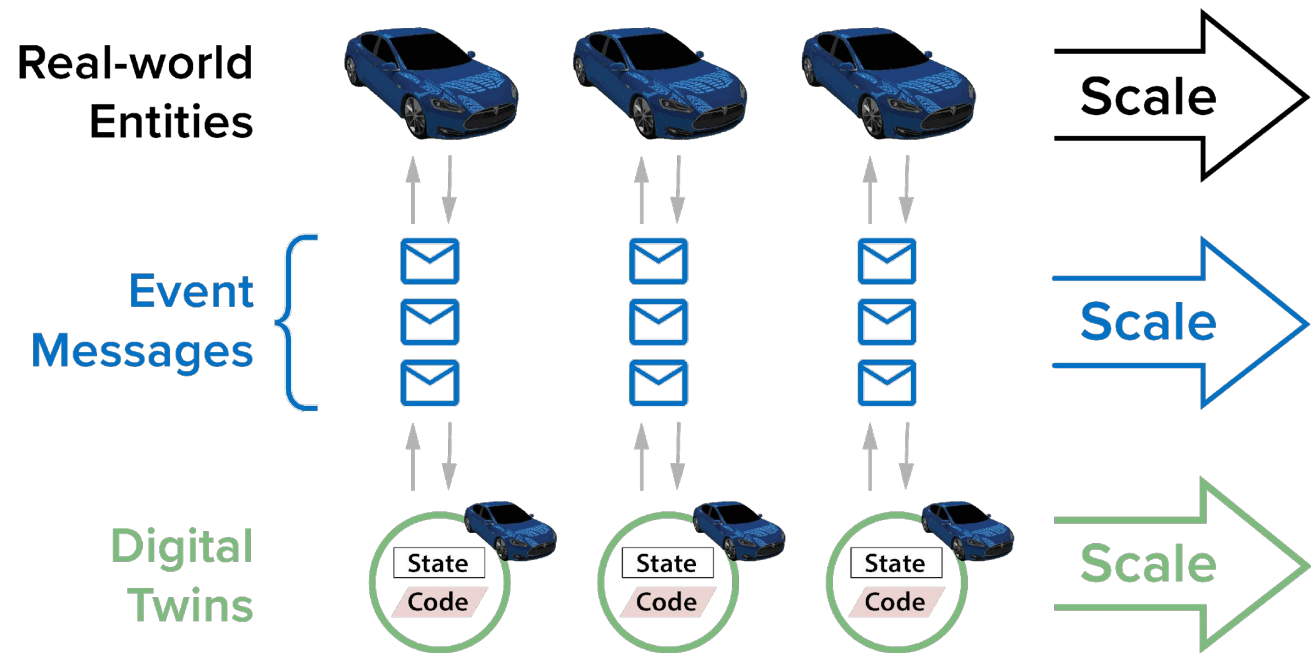
RTDTs Avoid Network Bottlenecks

- State-centric approach distributes events across state objects.
- Avoids network bottleneck accessing remote data store from event pipeline.
 - Network bottlenecks prevent scalable throughput.



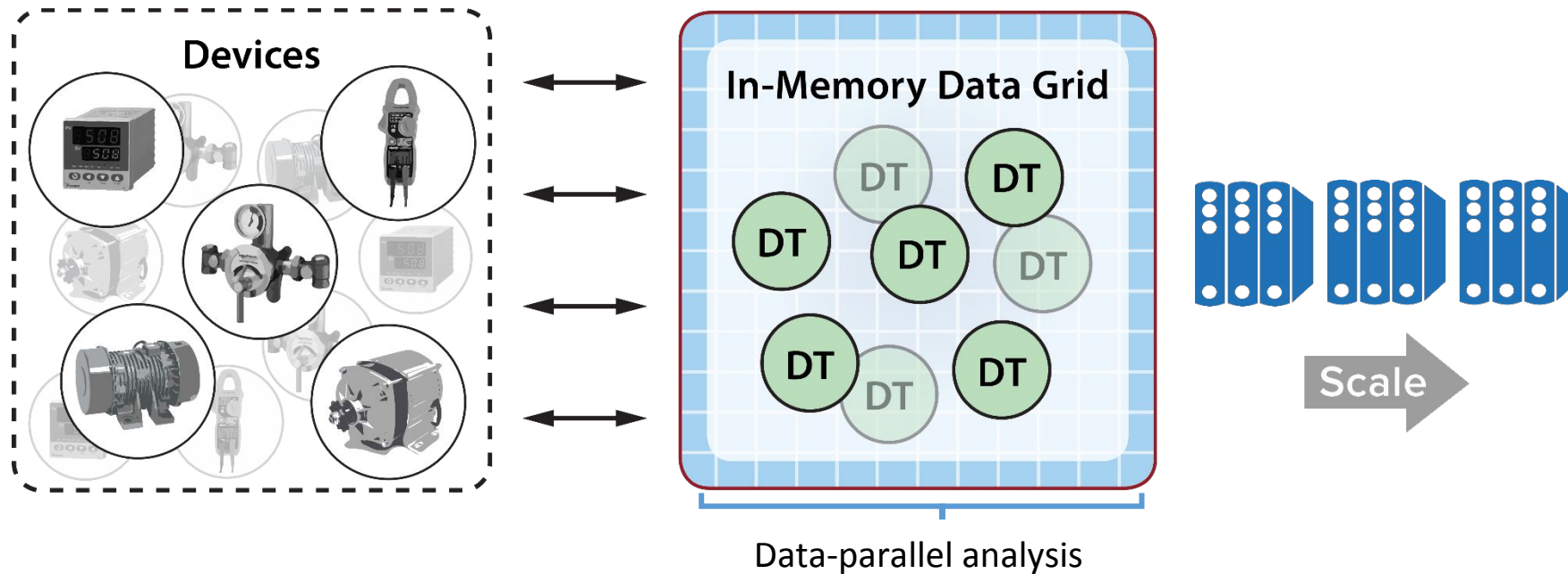
RTDTs Enable Fast, Scalable Performance

- When hosted on an in-memory computing platform, they can:
- Process event message in 1-3 milliseconds.
- Complete typical data-parallel analysis in ~5-10 seconds.
- Performance can transparently scale to handle 1M+ instances.



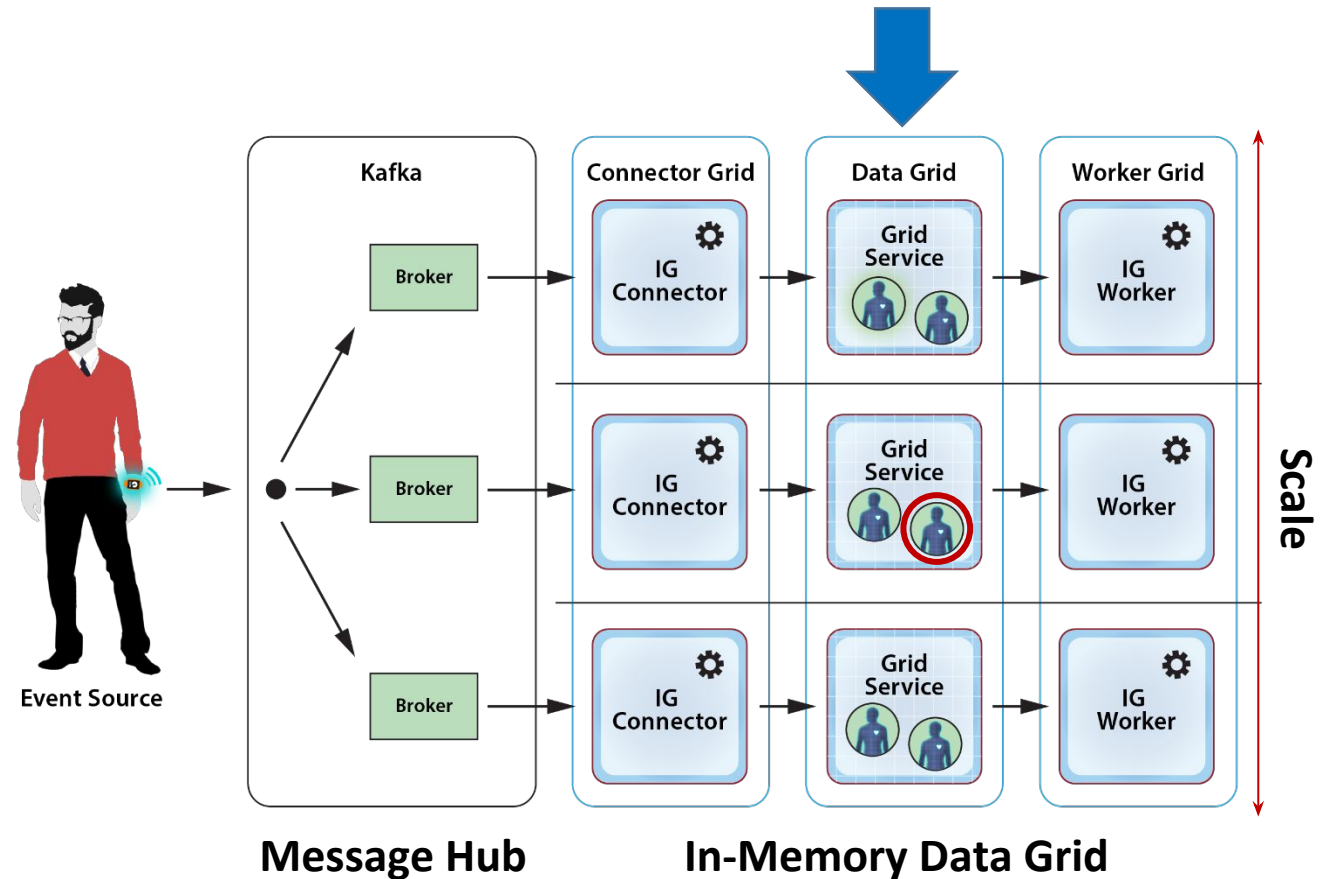
RTDTs Leverage In-Memory Computing

- State objects can be hosted within an in-memory data grid (IMDG).
- IMDG delivers event messages to state objects and runs message processor.
- IMDG can perform data-parallel analysis in place across state objects.



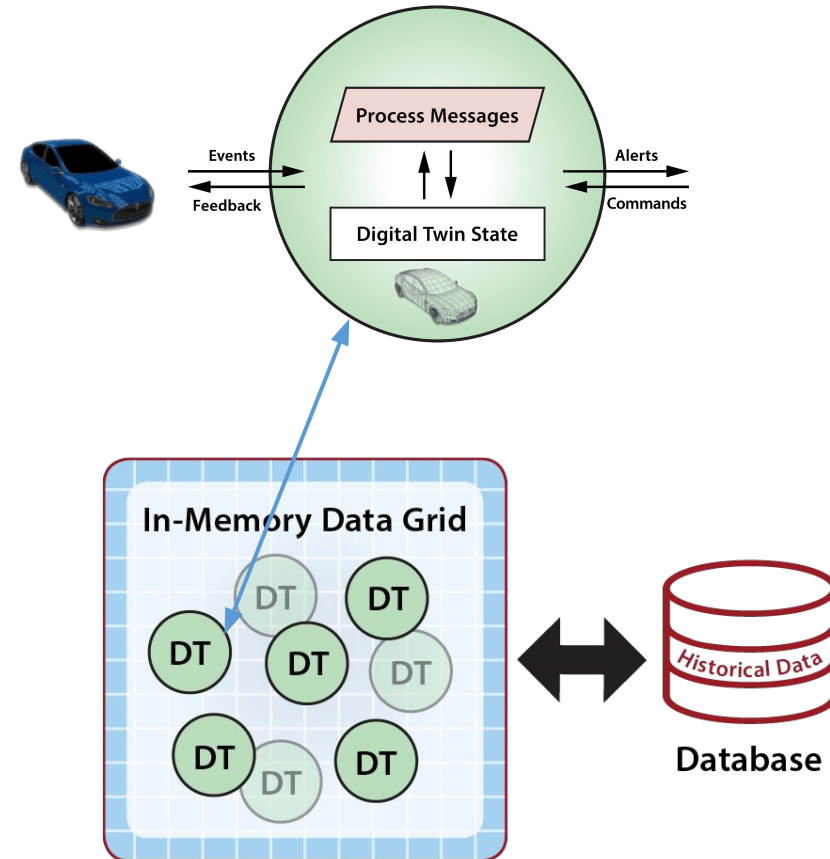
How an IMDG Runs Digital Twin Models

- Digital twin instances are hosted as objects in the Data Grid.
- Digital twin models run in an IG called the Worker Grid.
- Connectors run in an IG called the Connector Grid.
- Connectors invoke message processor on the server hosting the device's instance object.
 - Steers messages to object by id.
 - This minimizes network overhead.



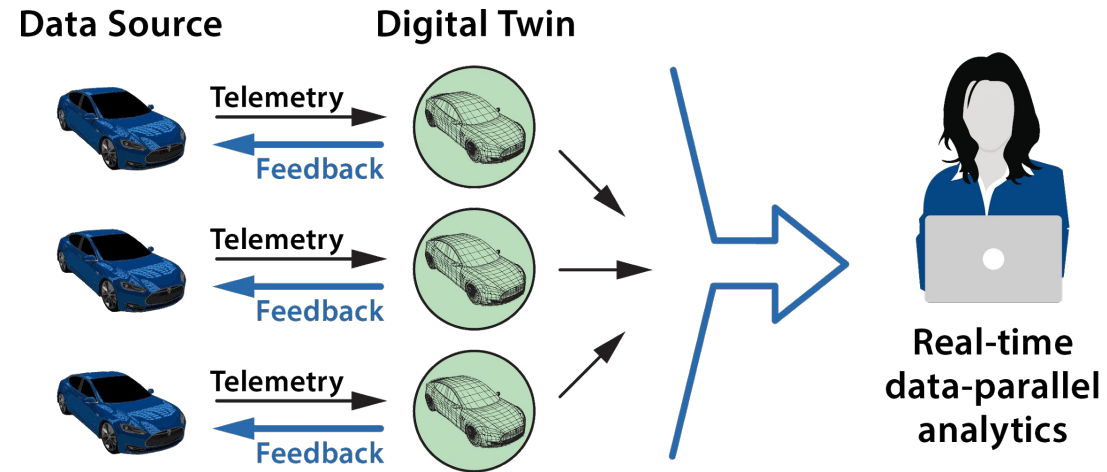
RTDTs Can Access Historical State

- Digital twins store dynamic state information in memory for fast access.
- Also can retrieve slowly- changing data from a database:
 - Device parameters
 - Maintenance history
- Can update database:
 - Event-message history
 - Significant changes to the device

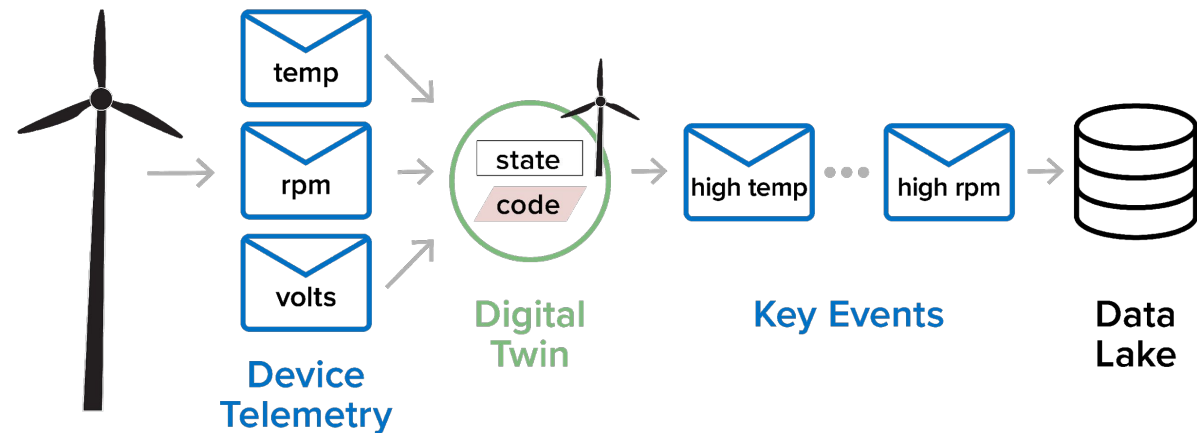


RTDTs Enable Telemetry Filtering

- Real-time digital twins create dynamic state information derived from incoming telemetry.
 - This information can be aggregated to spot patterns and trends.

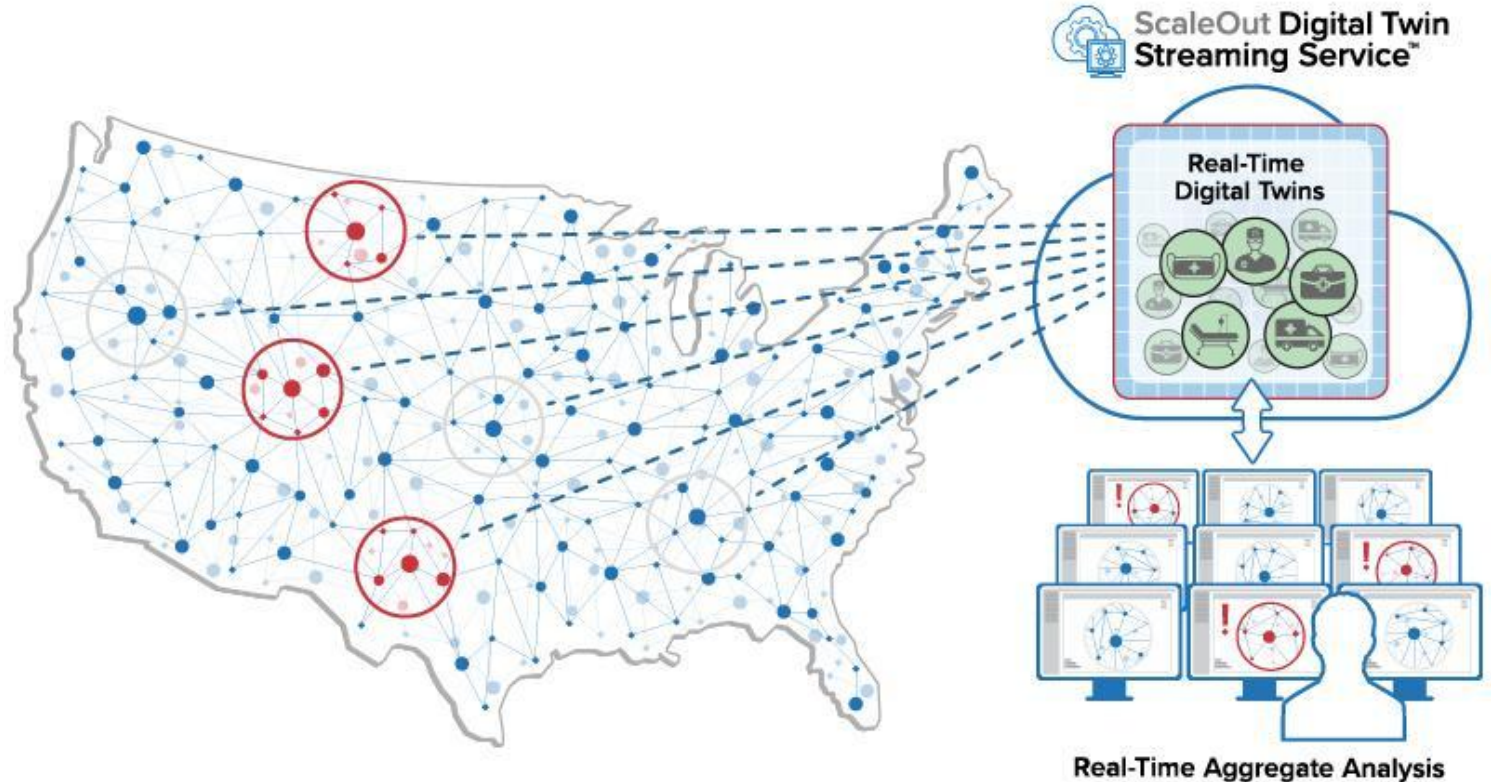


- Real-time digital twins also can filter events for offline analysis in the data lake:



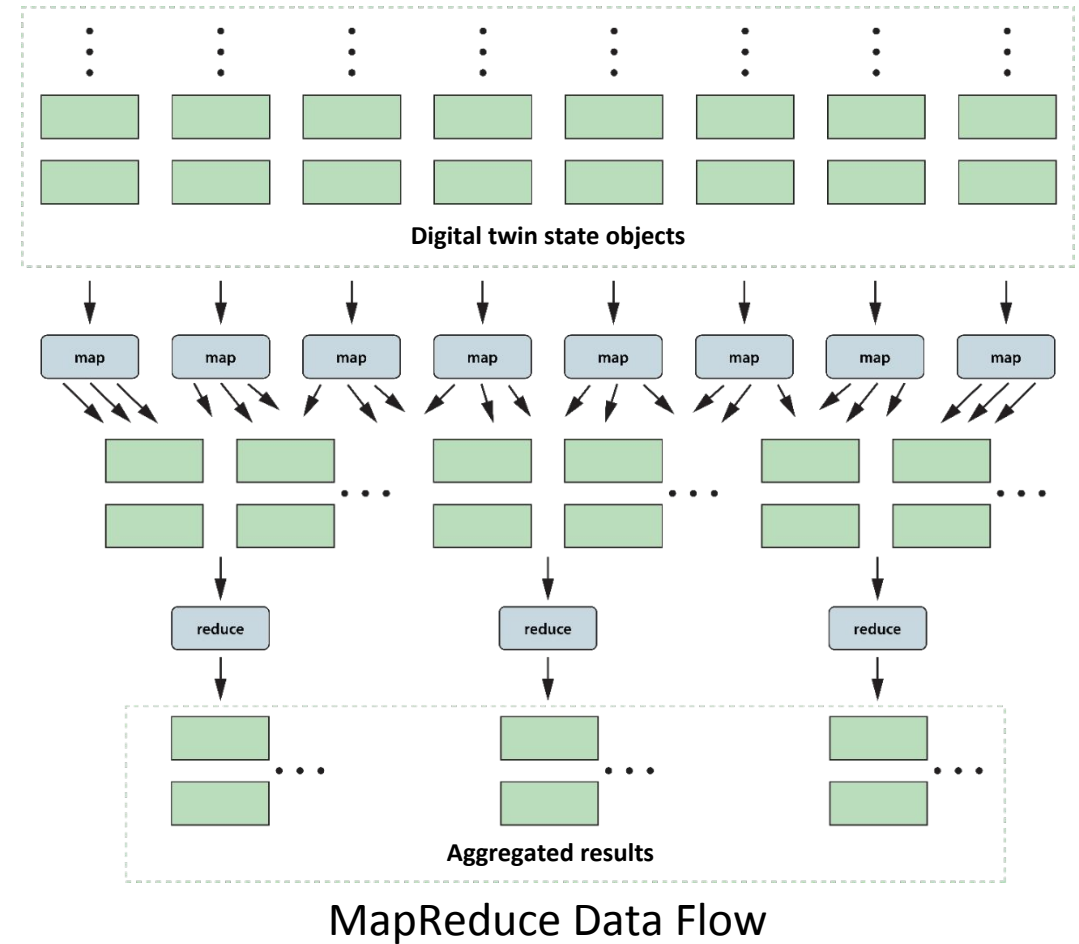
RTDTs Enable Aggregate Analysis

- **Simultaneously** analyzes all real-time digital twin instances to identify and visualize widespread trends in real time.
- Boosts **situational awareness**.
- Some examples:
 - Offer flash sale to individual online shoppers *and* tweak it in real time by tracking overall impact.
 - Manage wind turbines *and* spot preventive action.
 - Monitor power stations *and* identify rolling blackout.
 - Track a rental car fleet *and* alert unplanned road closure.



Aggregate Analysis with MapReduce

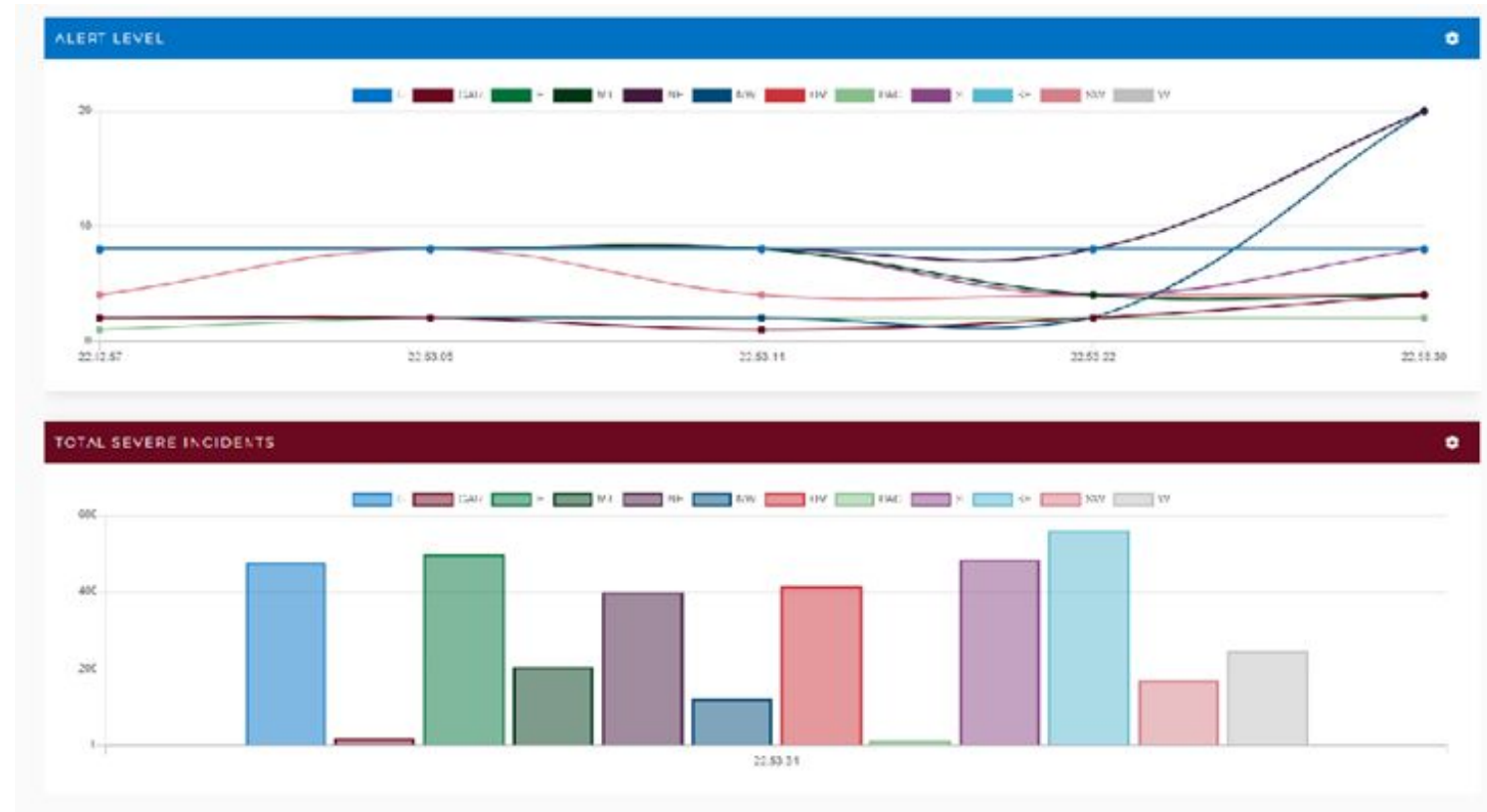
- A well-known, data-parallel technique:
- Aggregates property values across all instances of a model.
- Allows results to be grouped according to the value of another property.
 - Example: Ave. vehicle speed by county
- Runs seamlessly within an IMDG:
 - Runs concurrently with event processing.
 - Avoids network bottlenecks.
 - Avoids delay for offline processing.



Example Output of Aggregate Analysis

Aggregate analysis quickly pinpoints emerging issues for managers:

- Aggregates state properties in all RTDT instances and displays results.
- Can runs continuously as a background MapReduce job.

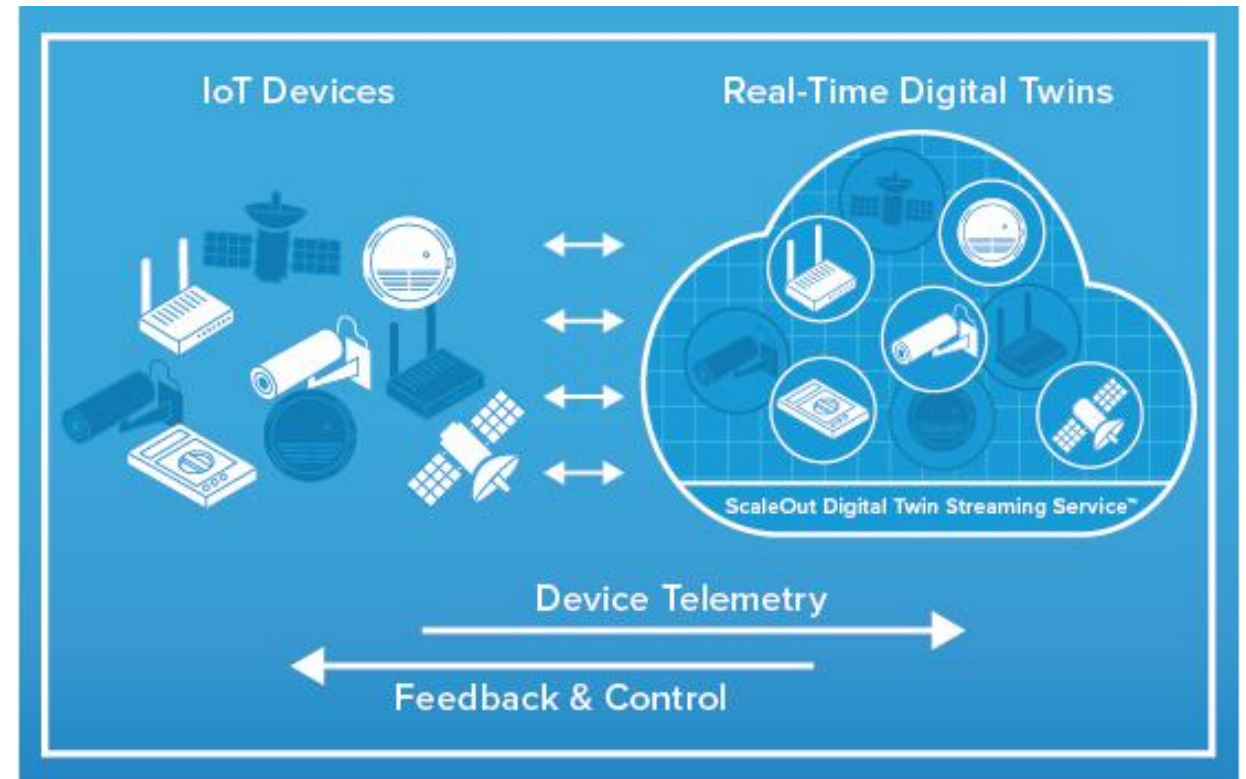


New technology for real-time monitoring and streaming analytics at scale



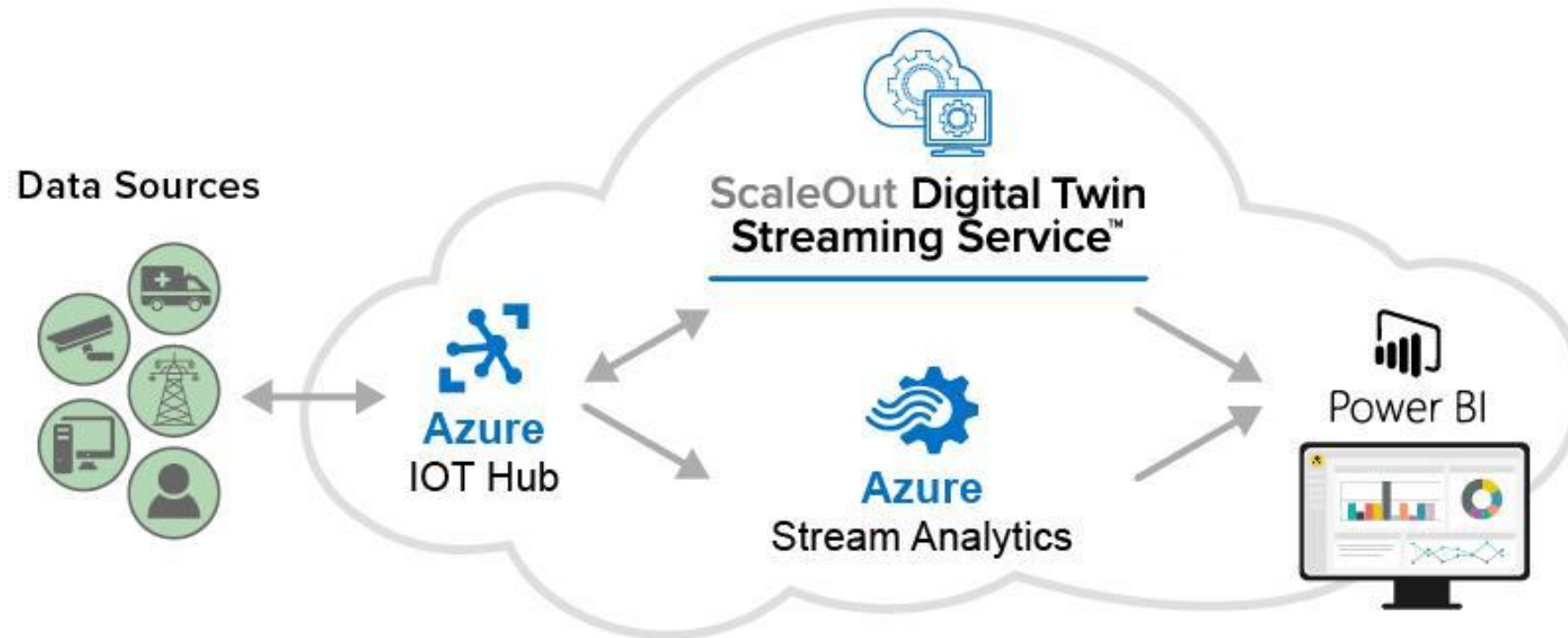
ScaleOut Digital Twin[™] Streaming Service

- Build & deploy real-time digital twin models
- Create & visualize real-time aggregate analytics
- Use as an Azure-hosted cloud service
- Intuitive web-based UI for ease of use
- Connectors for Azure, AWS, Kafka, and REST
- Backed by ScaleOut support



Integrates Into Cloud Infrastructure

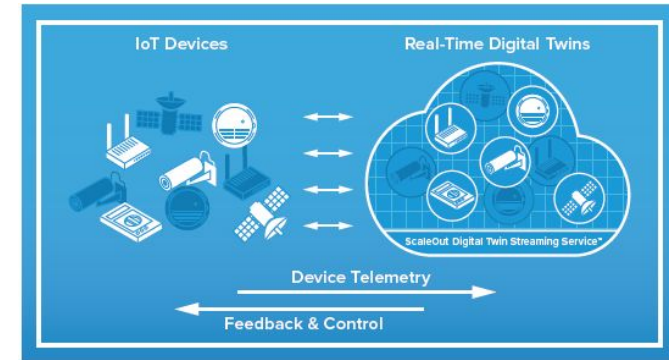
- Runs alongside other Azure streaming services:
 - Connects to Azure IoT Hub (and others) to exchange messages with data sources.
 - Optionally exports real-time aggregate statistics to Power BI for visualization.





ScaleOut Digital Twin Streaming Service™

- Build & deploy real-time digital twin models
- Create & visualize real-time aggregate analytics
- Use as a hosted service
- Intuitive web-based UI for ease of use
- Connectors for Azure, AWS, Kafka, and REST
- Backed by ScaleOut support

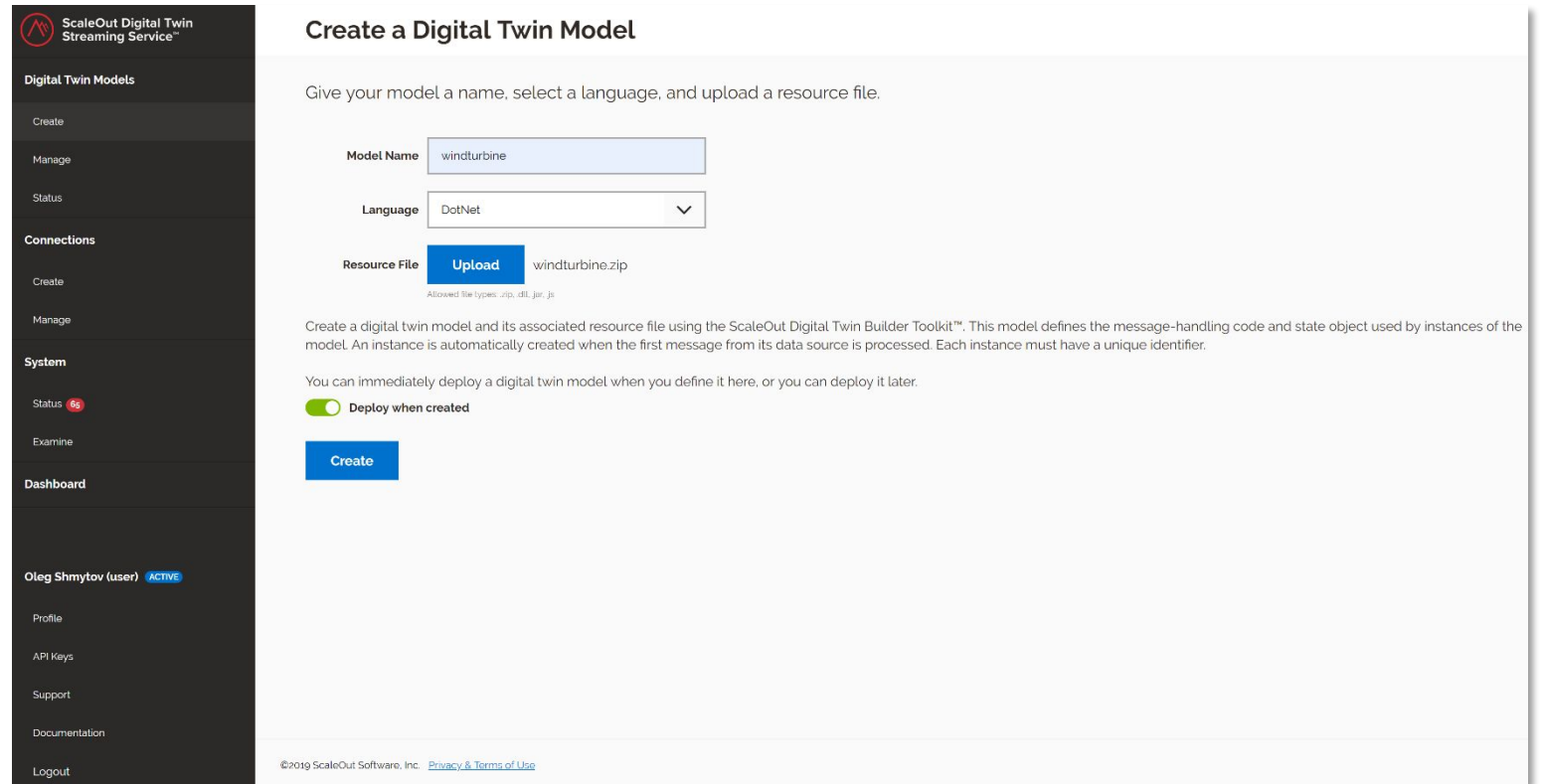


ScaleOut StreamServer™

- Run on-premises
- Deploy real-time digital twin models & connectors
- Use APIs to access all features

Deploying a Digital Twin to the Cloud

- Model is first created using APIs.
- UI uploads code from a zip file which contains dll's for all dependencies.
- UI selects language runtime, such as Java, C#, JavaScript.



ScaleOut Digital Twin Streaming Service™

Digital Twin Models

- Create
- Manage
- Status

Connections

- Create
- Manage

System

- Status 65
- Examine

Dashboard

Oleg Shmytov (user) ACTIVE

- Profile
- API Keys
- Support
- Documentation
- Logout

Create a Digital Twin Model

Give your model a name, select a language, and upload a resource file.

Model Name

Language

Resource File windturbine.zip

Allowed filetypes: zip, dll, jar, js

Create a digital twin model and its associated resource file using the ScaleOut Digital Twin Builder Toolkit™. This model defines the message-handling code and state object used by instances of the model. An instance is automatically created when the first message from its data source is processed. Each instance must have a unique identifier.

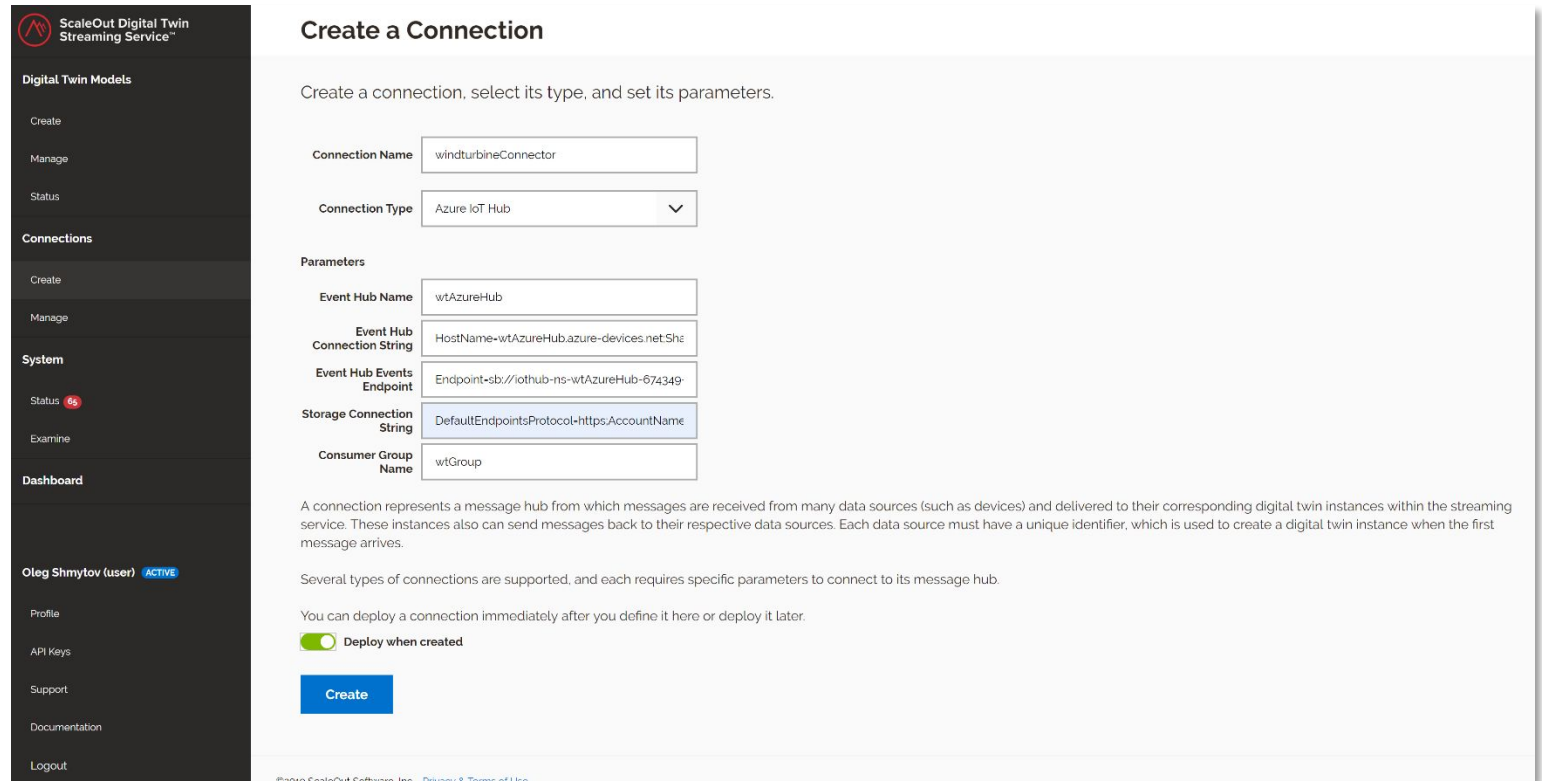
You can immediately deploy a digital twin model when you define it here, or you can deploy it later.

☒ Deploy when created

©2019 ScaleOut Software, Inc. [Privacy & Terms of Use](#)

Deploying a Connector to the Cloud

- Connectors can be created by specifying the hub type and connection parameters.
- Hub types include:
 - Azure IoT Hub
 - AWS Core
 - Azure Kafka
- REST is accessed using a URL.



The screenshot shows the 'Create a Connection' interface of the ScaleOut Digital Twin Streaming Service. On the left is a dark sidebar with navigation links: Digital Twin Models (Create, Manage, Status), Connections (Create, Manage), System (Status, Examine), and Dashboard. The main content area is titled 'Create a Connection' and includes instructions: 'Create a connection, select its type, and set its parameters.' The form fields are: Connection Name (windturbineConnector), Connection Type (Azure IoT Hub), Event Hub Name (wtAzureHub), Event Hub Connection String (HostName=wtAzureHub.azure-devices.net;Sh...), Event Hub Events Endpoint (Endpoint=sb://iothub-ns-wtAzureHub-674349-...), Storage Connection String (DefaultEndpointsProtocol=https;AccountName...), and Consumer Group Name (wtGroup). Below the form, there is explanatory text about connections and a 'Deploy when created' toggle switch. A blue 'Create' button is at the bottom.

Create a Connection

Create a connection, select its type, and set its parameters.

Connection Name: windturbineConnector

Connection Type: Azure IoT Hub

Parameters

Event Hub Name: wtAzureHub

Event Hub Connection String: HostName=wtAzureHub.azure-devices.net;Sh...

Event Hub Events Endpoint: Endpoint=sb://iothub-ns-wtAzureHub-674349-

Storage Connection String: DefaultEndpointsProtocol=https;AccountName

Consumer Group Name: wtGroup

A connection represents a message hub from which messages are received from many data sources (such as devices) and delivered to their corresponding digital twin instances within the streaming service. These instances also can send messages back to their respective data sources. Each data source must have a unique identifier, which is used to create a digital twin instance when the first message arrives.

Several types of connections are supported, and each requires specific parameters to connect to its message hub.

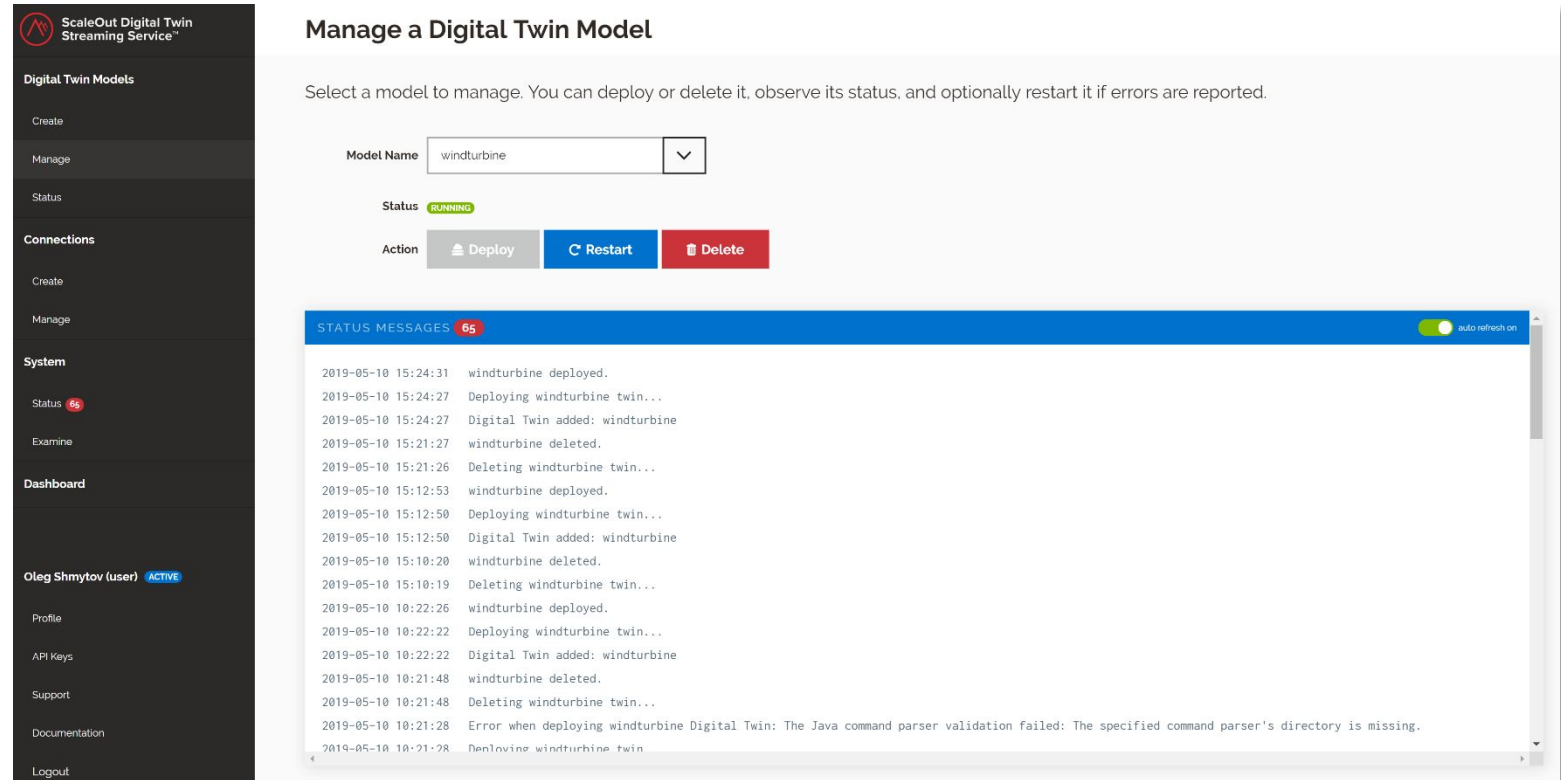
You can deploy a connection immediately after you define it here or deploy it later.

☒ Deploy when created

Create

Managing Digital Twin Models in the Cloud

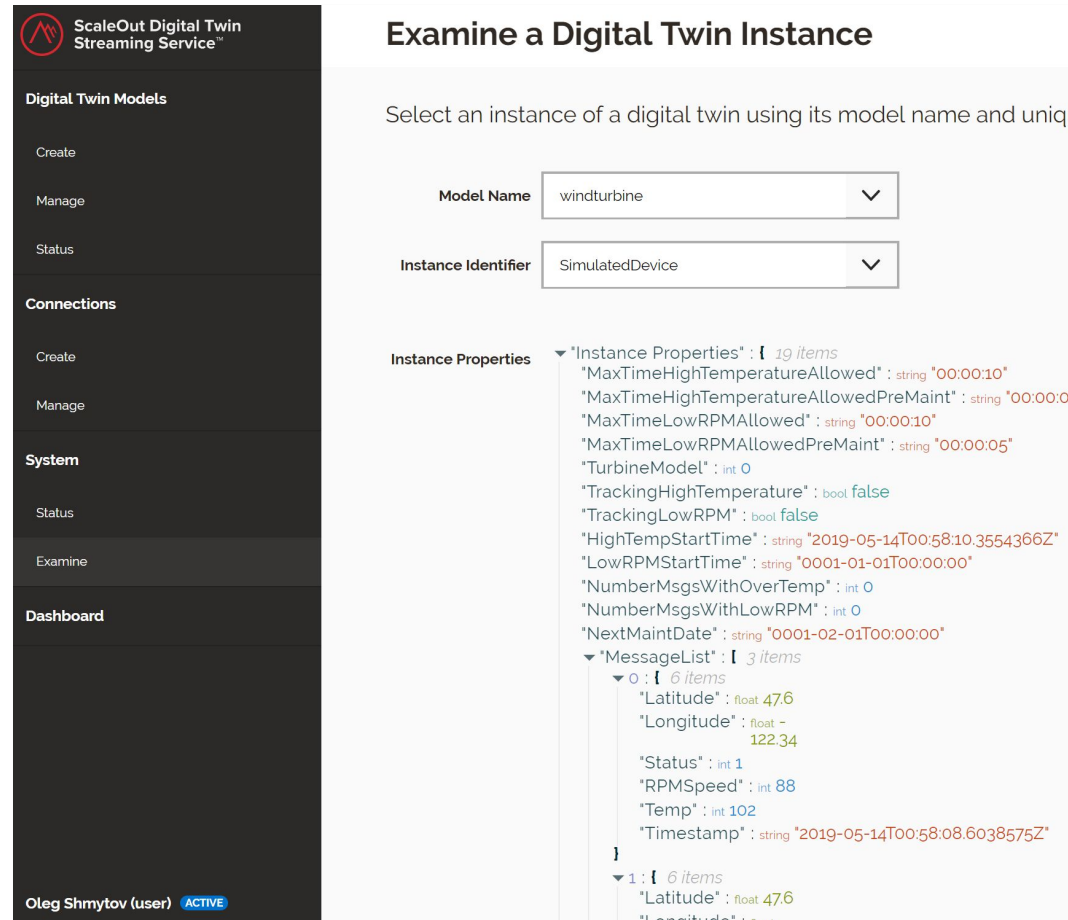
- Each model can be independently managed to check status and restart as necessary:



The screenshot displays the ScaleOut Digital Twin Streaming Service interface. On the left is a dark sidebar with navigation links: Digital Twin Models (Create, Manage, Status), Connections (Create, Manage), System (Status 64, Examine), Dashboard, and user information (Oleg Shmytov (user) ACTIVE, Profile, API Keys, Support, Documentation, Logout). The main panel is titled 'Manage a Digital Twin Model' and includes instructions: 'Select a model to manage. You can deploy or delete it, observe its status, and optionally restart it if errors are reported.' Below this, a 'Model Name' dropdown is set to 'windturbine'. The 'Status' is 'RUNNING' in a green box. The 'Action' bar contains 'Deploy', 'Restart', and 'Delete' buttons. A 'STATUS MESSAGES' section with 65 messages is shown, with an 'auto refresh on' toggle. The messages log shows a sequence of events for the 'windturbine' model, including deployment, addition, deletion, and a final error message: 'Error when deploying windturbine Digital Twin: The Java command parser validation failed: The specified command parser's directory is missing.'

Examining a Digital Twin Instance

- The properties for each digital twin instance (i.e., for each device) can be examined:



The screenshot displays the ScaleOut Digital Twin Streaming Service interface. On the left is a dark sidebar with navigation options: Digital Twin Models, Connections, System, and Dashboard. The main panel is titled 'Examine a Digital Twin Instance' and contains a form to select a digital twin instance by Model Name and Instance Identifier. Below the form, the 'Instance Properties' are listed in a JSON-like format, showing various attributes such as temperature, RPM, and status for a simulated device.

ScaleOut Digital Twin Streaming Service

Digital Twin Models

- Create
- Manage
- Status

Connections

- Create
- Manage

System

- Status
- Examine

Dashboard

Oleg Shmytov (user) **ACTIVE**

Examine a Digital Twin Instance

Select an instance of a digital twin using its model name and unique identifier

Model Name windturbine

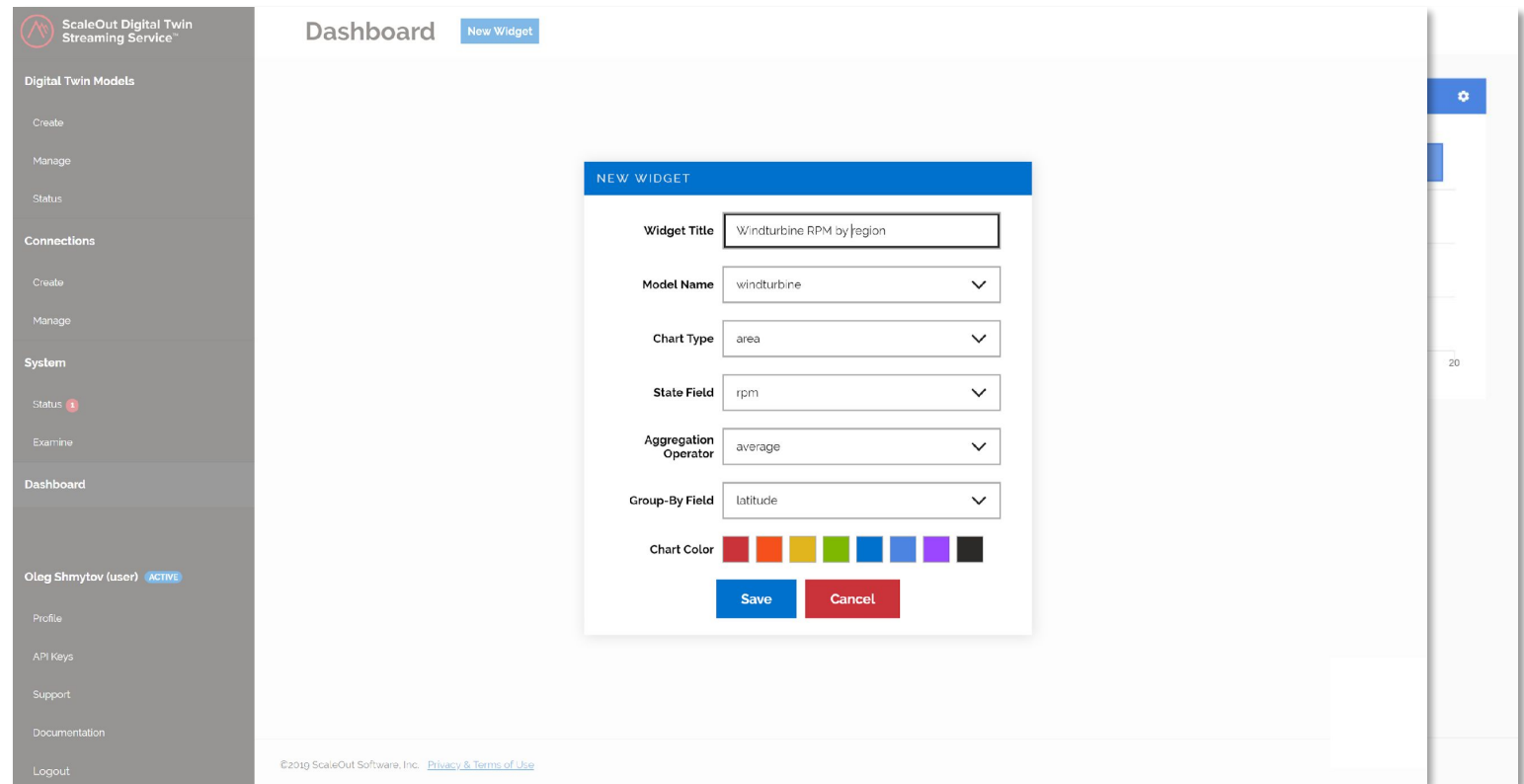
Instance Identifier SimulatedDevice

Instance Properties

```
{
  "Instance Properties": {
    "MaxTimeHighTemperatureAllowed": "00:00:10",
    "MaxTimeHighTemperatureAllowedPreMaint": "00:00:05",
    "MaxTimeLowRPMAllowed": "00:00:10",
    "MaxTimeLowRPMAllowedPreMaint": "00:00:05",
    "TurbineModel": 0,
    "TrackingHighTemperature": false,
    "TrackingLowRPM": false,
    "HighTempStartTime": "2019-05-14T00:58:10.3554366Z",
    "LowRPMStartTime": "0001-01-01T00:00:00",
    "NumberMsgsWithOverTemp": 0,
    "NumberMsgsWithLowRPM": 0,
    "NextMaintDate": "0001-02-01T00:00:00",
    "MessageList": [
      {
        "Latitude": 47.6,
        "Longitude": -122.34,
        "Status": 1,
        "RPMSpeed": 88,
        "Temp": 102,
        "Timestamp": "2019-05-14T00:58:08.6038575Z"
      },
      {
        "Latitude": 47.6,
        "Longitude": -122.34,
        "Status": 1,
        "RPMSpeed": 88,
        "Temp": 102,
        "Timestamp": "2019-05-14T00:58:08.6038575Z"
      }
    ]
  }
}
```


Collecting Aggregate Statistics

- “Widgets” can be created for digital twin models to display aggregate statistics:
- Performs periodic MapReduce on selected state properties.
- Runs every few seconds.



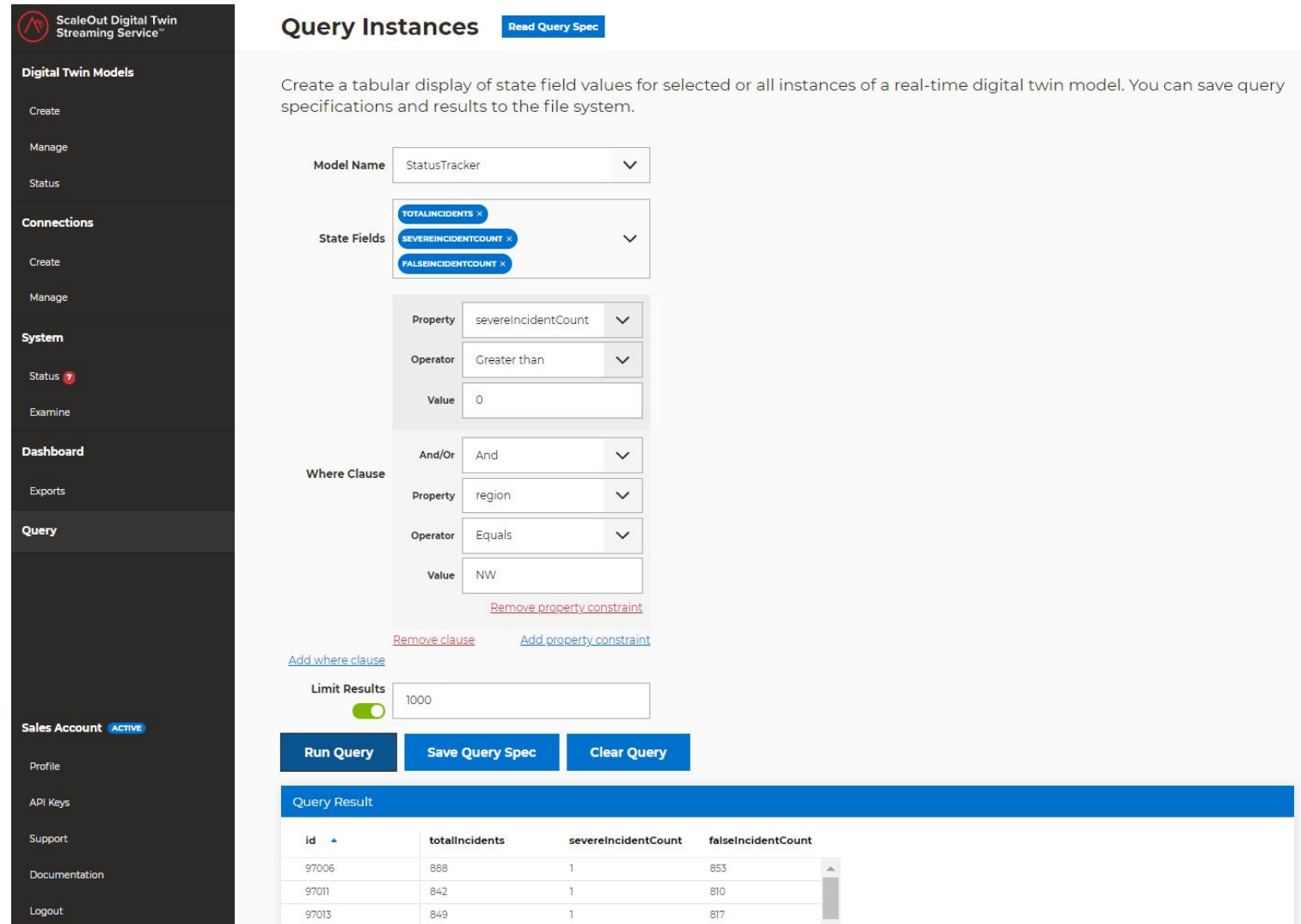
The screenshot displays the ScaleOut Digital Twin Streaming Service interface. On the left is a sidebar menu with sections: Digital Twin Models (Create, Manage, Status), Connections (Create, Manage), System (Status, Examine), and Dashboard. The main area is titled 'Dashboard' and contains a 'New Widget' button. A modal window titled 'NEW WIDGET' is open, showing the following configuration fields:

- Widget Title: Windturbine RPM by region
- Model Name: windturbine
- Chart Type: area
- State Field: rpm
- Aggregation Operator: average
- Group-By Field: latitude
- Chart Color: A row of color swatches (red, orange, yellow, green, blue, purple, black).

At the bottom of the modal are 'Save' and 'Cancel' buttons. The footer of the dashboard shows '©2019 ScaleOut Software, Inc. Privacy & Terms of Use'.

Querying Real-Time Digital Twin Instances ScaleOut Software

- Instances can be queried using a simplified, UI-based version of SQL SELECT.
- The number of records returned can be limited.
- Runs in a few seconds.



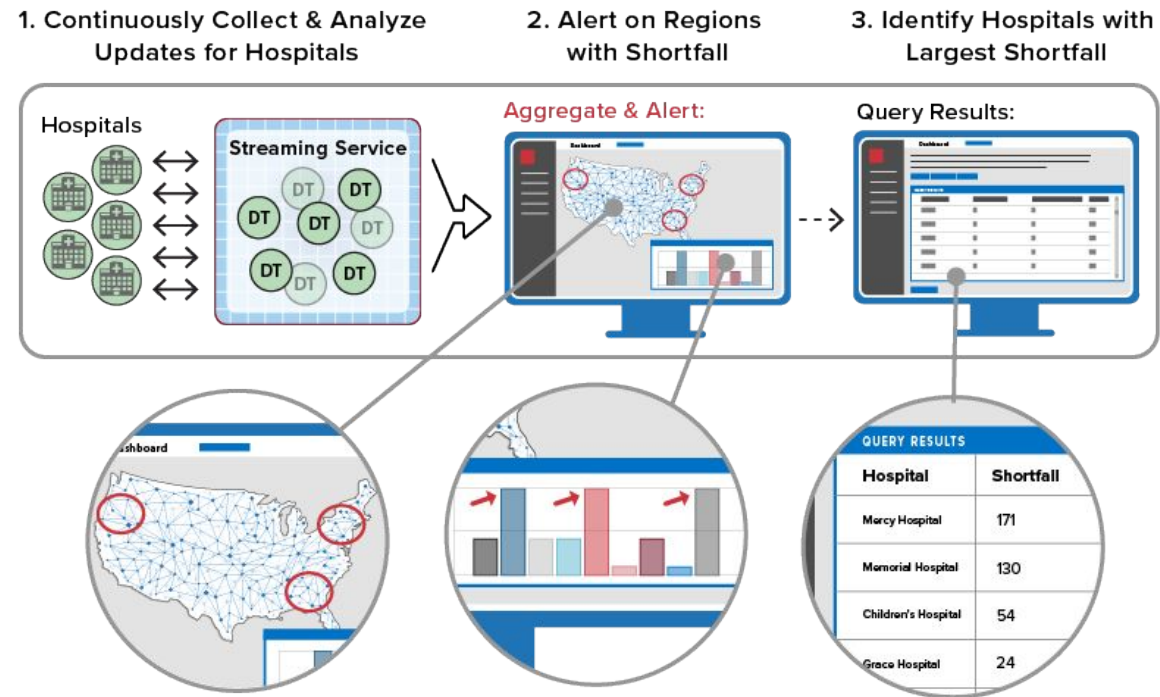
The screenshot displays the 'Query Instances' interface of the ScaleOut Digital Twin Streaming Service. The left sidebar contains navigation links for Digital Twin Models, Connections, System, Dashboard, and Query. The main panel shows a query configuration form for the 'StatusTracker' model. It includes fields for State Fields (TOTALINCIDENTS, SEVEREINCIDENTCOUNT, FALSEINCIDENTCOUNT), a Where Clause (severeIncidentCount Greater than 0, region Equals NW), and a Limit Results toggle set to 1000. Below the form are buttons for 'Run Query', 'Save Query Spec', and 'Clear Query'. The 'Query Result' table shows three rows of data.

id	totalIncidents	severeIncidentCount	falseIncidentCount
97006	888	1	853
97011	842	1	810
97013	849	1	817

Using Aggregate Analytics & Query

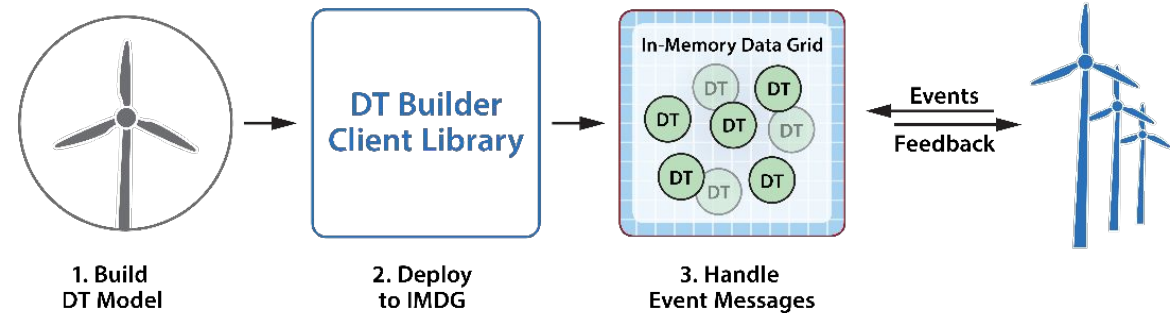
Maximizes situational awareness.

- Integrated analytics engine combines key RTDT data in seconds.
 - Example: Determine largest shortfall in hospital supplies by region.
- Streaming service lets users visualize results.
 - Example: Show shortfall by region as a bar chart to alert on problem areas as they occur.
- Users query RTDT data to identify issues and take action.
 - Example: Query RTDTs to find specific hospitals with largest shortfall in affected regions.

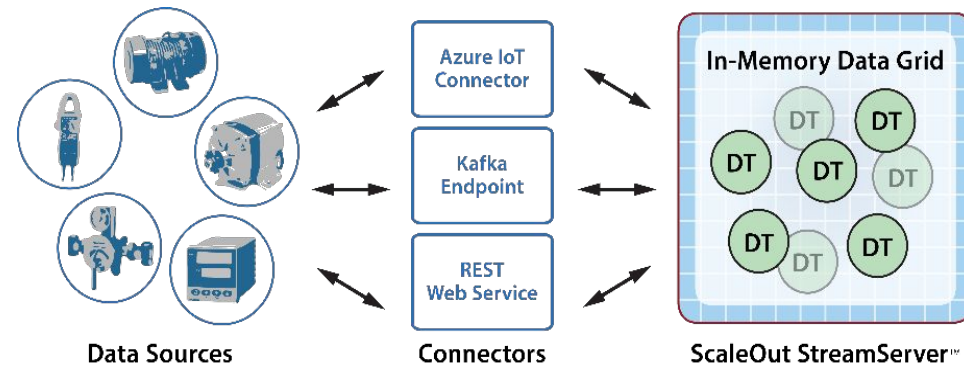


Building and Deploying Real-Time Digital Twins

- **Step 1:** Build a real-time digital twin model and deploy to the cloud service for execution:



- **Step 2:** Connect the IMDG to a message hub (e.g., Azure IoT Hub, AWS IoT, Kafka, REST, etc.):
 - RTDT instances are automatically created as messages flow in from data sources.



Why Use Specific APIs for Digital Twins?

- **Simplifies application design**; avoids complexity of underlying IMDG APIs, including:
 - Explicitly managing and accessing state objects in the IMDG
 - Orchestrating the staging of message-processing code across the IMDG
 - Connecting digital twins to data sources
 - Delivering messages to digital twins and back to data sources
 - Ensuring highly available message handling
- Digital twin APIs and services allow the application to focus on:
 - Defining message-processing code for each type of data source
 - Defining the dynamic state information to be managed for each data source
 - Describing periodic data-parallel analytics to be performed across all digital twins of a given type

- Application defines a message type for incoming messages.
- Application implements a base class `DigitalTwinBase` to define a state object that holds instance properties and optional event lists.
- Application implements a message processor method:

`ProcessMessage(stateObject, processingContext, messageList)`

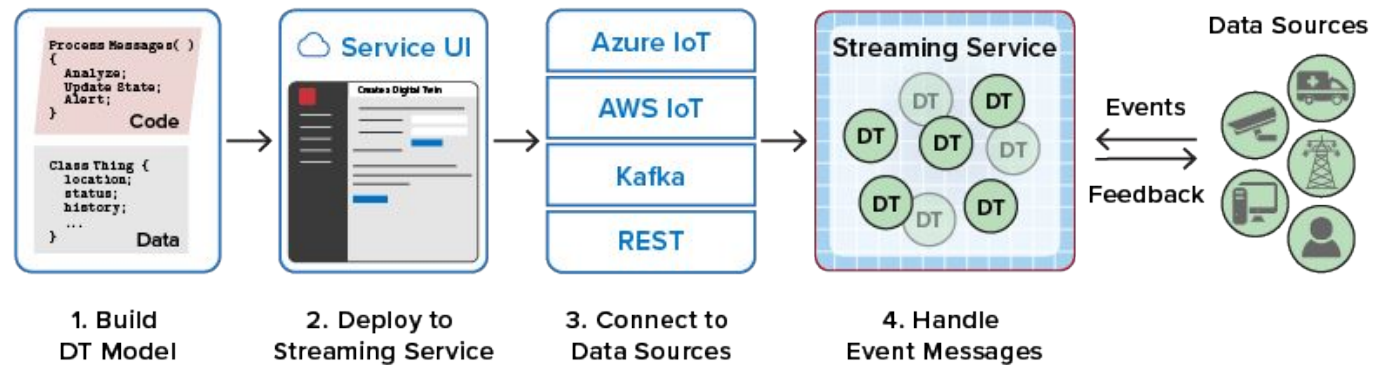
- `ProcessingContext` defines APIs for sending messages to data source or to other twins.
- Message list contains an enumerable list of messages that arrived since last call to `ProcessMessage`.
 - Hides latency by handling multiple messages at once.
 - Enables single acknowledgment for a group of messages.

Deployment APIs

- Deploy model to IMDG:

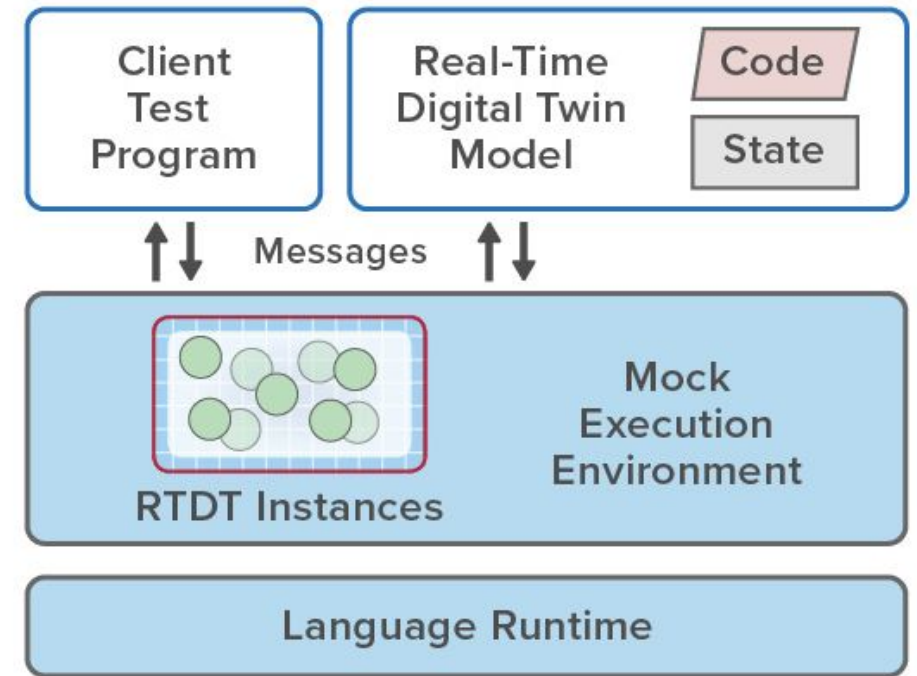
```
builder = new ModelBuilder()  
    .AddDependency("code.dll")  
    .AddModel<stateObjectType,  
        messageProcessorType,  
        eventMessageType>()  
    .Build();
```

- Deploy model to the cloud service using the UI:



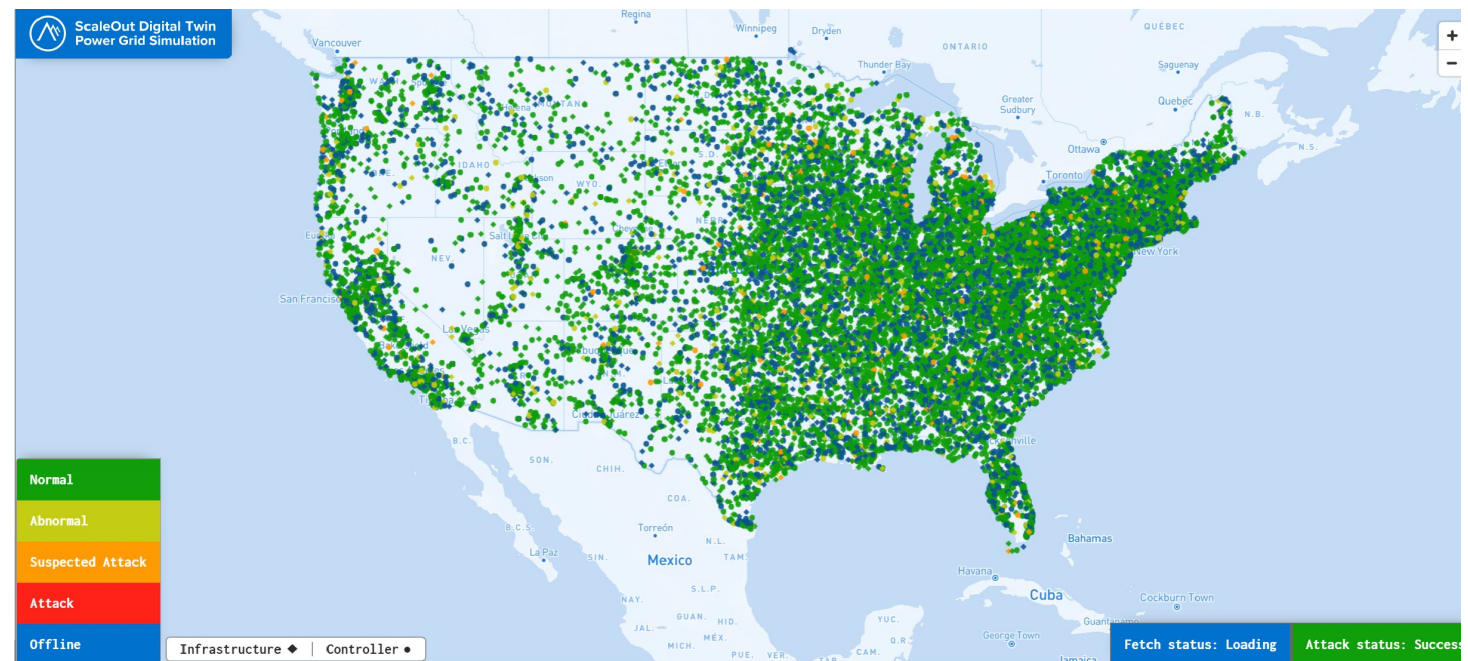
Development in a Mock Environment

- Debugging and test within the cloud service can be challenging.
 - Cannot set breakpoints and examine state.
 - Difficult to verify all code paths.
- Mock environment allows development and test on developer's workstation:
 - Uses same base classes as production deployment.
 - Replaces `ProcessingContext` with mock version.
 - Deploys model with same APIs as used on-premises.
 - Allows sending/receiving messages from instances.
 - Allows instance state to be examined.



Example: Simulated Power Grid

- **Goal: Demonstrate value of enhanced situational awareness**
- **Overview:**
 - Simulates 20K nodes within a power grid spanning the US and subject to outages or attack.
 - Each node streams telemetry with its status to its real-time digital twin in the cloud service.
 - Real-time twin evaluates telemetry and updates derived state: *alert level*:
 - Assesses threat level for that node (range 0-20)
 - Updated by examining telemetry with knowledge of node's function and reporting history (e.g., false positives)



Fast development and deployment:

- Developer just implements one message processing method and state object for a single real-time digital twin.
 - Approximately 20 Java statements in this demonstration
- Platform automatically runs this code for all data sources.
 - Automatically correlates incoming messages by data source.
 - Automatically scales to handle large workloads.
- Developer avoids:
 - Selecting telemetry out of a single, combined event stream
 - Creating ad hoc storage for contextual data
 - Pushing the data into a data lake for offline analysis using Spark

```
Process Messages( )  
{  
    Analyze;  
    Update State;  
    Alert;  
}
```

Code

```
Class Tower {  
    location;  
    status;  
    history;  
    ...  
}
```

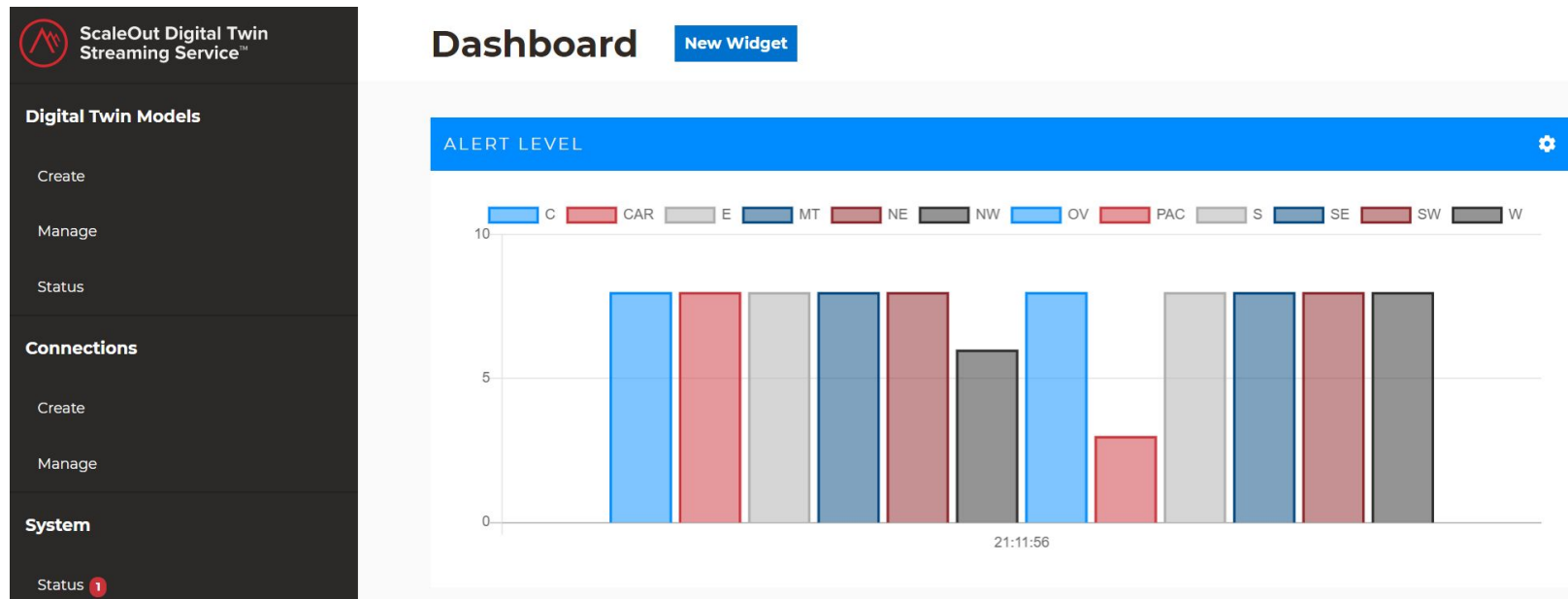
Data

Streaming Service Application Code

Example: Simulated Power Grid

Aggregate analysis by the cloud service quickly pinpoints emerging threats for managers:

- Evaluates alert level in digital twin state for all nodes and aggregates results.
- Dashboard widget displays maximum alert level by region.
- Runs every 5 seconds as a background MapReduce job.



- Real-time stream-processing does not allow thousands of data sources to be individually tracked and does not have integrated aggregate real-time analytics.
- Traditional approaches (Lambda Architecture, pipelines) limit real-time processing and don't perform aggregate analysis in real time.
- Real-time digital twins offer a breakthrough:
 - Deeper introspection in real time
 - Simplified application design
 - Fast, scalable performance
- Enable vastly improved **situational awareness** and **response**.
- ScaleOut's in-memory data grid and cloud service provides a fast, scalable and easy to use execution platform.



ScaleOut Digital Twin
Streaming Service™

- **Session 1:** Technology Overview (Bill Bain, 1 hour)
 - What are Real-Time Digital Twins?
 - A Tour of the ScaleOut Digital Twin Streaming Service™
 - ScaleOut Digital Twin Builder™ Software Toolkit
 - Demo
- **Session 2:** Walkthrough: Building a Java Digital Twin for Security Tracking (Brandon Ripley, 45 min)
- Break (10 minutes)
- **Session 3:** Walkthrough: Building a C# Digital Twin for a Wind Turbine (Oleg Shmytov, 45 min)
- **Demo:** A Contact Tracing Application (Olivier Tritschler, 15 min)
- **Session 4:** Build Your First Real-Time Digital Twin (Team, 45 min)
- **Wrap-Up** (Bill Bain, 5 min)

- **Session 1:** Technology Overview (Bill Bain, 1 hour)
 - What are Real-Time Digital Twins?
 - A Tour of the ScaleOut Digital Twin Streaming Service™
 - ScaleOut Digital Twin Builder™ Software Toolkit
 - Demo
- **Session 2:** Walkthrough: Building a Java Digital Twin for Security Tracking (Brandon Ripley, 45 min)
- Break (10 minutes)
- **Session 3:** Walkthrough: Building a C# Digital Twin for a Wind Turbine (Oleg Shmytov, 45 min)
- **Demo:** A Contact Tracing Application (Olivier Tritschler, 15 min)
- **Session 4: Build Your First Real-Time Digital Twin (Team, 45 min)**
- **Wrap-Up** (Bill Bain, 5 min)

Natural Gas Smart Meter Digital Twin Model ScaleOut Software

Natural gas typically consists of methane (or ethane), propane and other elements. This gas is an extremely fast-acting, toxic substance. The maximum allowable short-term exposure level is 50 ppm for 15 minutes



Programming Exercise:

1. Create a digital twin model to process telemetry from a gas sensor for tracking gas concentration and alerting when maximum exposure has been reached.
2. Create messages which report the id, current ppm, and timestamp for a gas sensor.
3. Set an alert flag when either the gas level exceeds 50 ppm for more than 15 minutes or the level spikes to 200 ppm.
4. (Bonus task): in addition to setting the flag add the code for sending a pipe shutdown control message back to the meter device using the `SendToDataSource` method.



ScaleOut Software

Assessing Threats for the Power Grid

IMCS Training Session: Java Demo

October 27, 2020

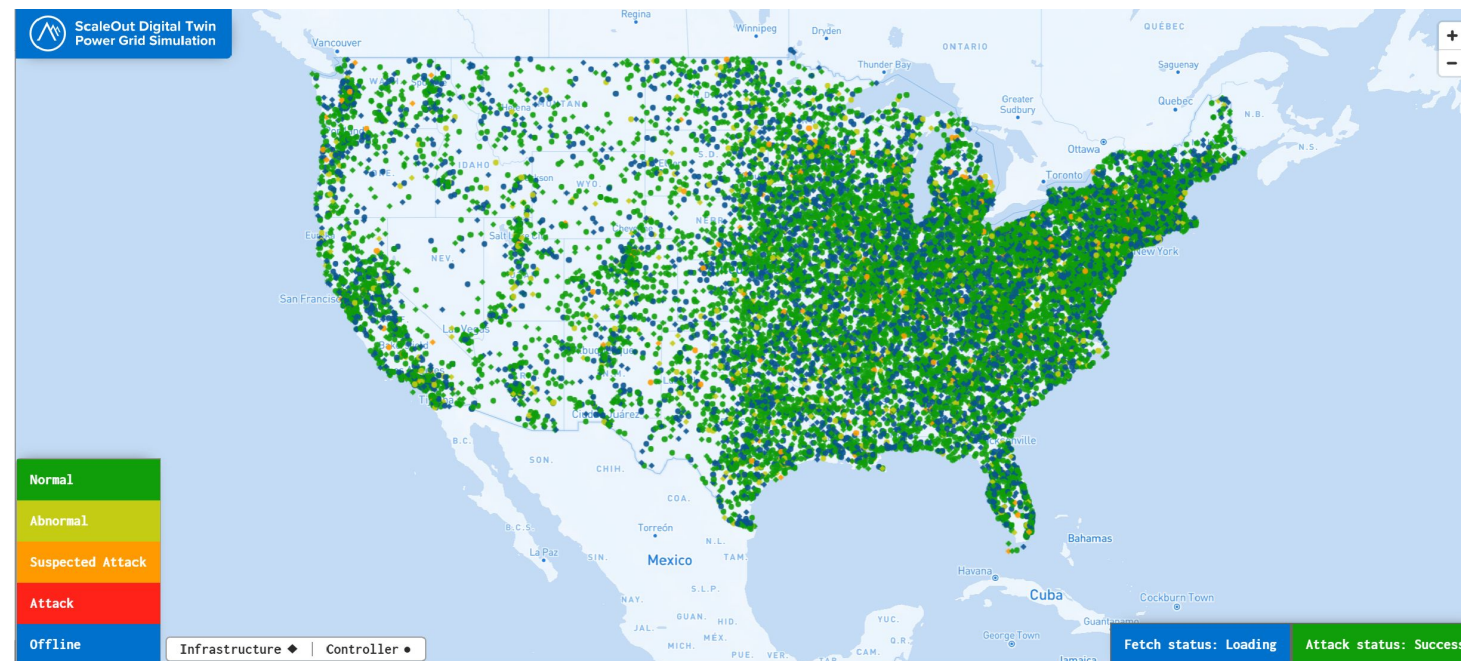
Brandon Ripley, Senior Software Engineer

brandonr@scaleoutsoftware.com

- Introduction and Session Overview
 - Where to Go for Help
 - Community Forum
 - User's Guide, Project Configuration Guide
 - API Documentation
 - Java Digital Twin API Overview
 - Power Grid Code Example
 - Wrap-up and Questions
-

Example: Simulated Power Grid

- **Goal: Demonstrate value of enhanced situational awareness**
- **Overview:**
 - Simulates 20K nodes within a power grid spanning the US and subject to outages or attack.
 - Each node streams telemetry with its status to its real-time digital twin in the cloud service.
 - Real-time twin evaluates telemetry and updates derived state: *alert level*:
 - Assesses threat level for that node (range 0-20)
 - Updated by examining telemetry with knowledge of node's function and reporting history (e.g., false positives)



Challenge

- Static State is Not Enough
- Many Data Sources
- Real-time Messaging

Solution – Real-Time Digital Twin Model

- Derived, Operational State
 - Live Data Source Tracking
 - Real-time Responsiveness
-

Create the StatusTracker Model

1. Digital Twin Base Class
 - Assign State Variables
2. Message class
 - Represent DataSource Telemetry
3. MessageProcessor
 - Insert Business Logic

- Test Model with MockEnvironment
- Create Configuration
 - Called “model.json”
 - Specify the Types – DigitalTwin, Message, and MessageProcessor
- Package
 - Create Zip File with “model.json”, Model JAR, and other Dependency JARs
- Deploy
 - Cloud-Service or On-premises Deployment

Demo

StatusTracker Digital Twin Model

Available at:

[GitHub.com/scaleoutsoftware/DigitalTwinDemos](https://github.com/scaleoutsoftware/DigitalTwinDemos)

Conclusion

- Showed Useful Web Locations
- Overview of the Java Digital Twin Model
- Created StatusTracker Digital Twin Model
- Tested the StatusTracker with MockEnvironment API
- Deployed the StatusTracker to the Cloud-Service

- [Community Forum](#)
- [Real-Time Digital Twin Cloud Service User's Guide](#)
- [Java API Documentation](#)
- [Java Project Configuration Guide](#)
- [Maven Repository](#)
- [Digital Twin Model Core API](#)
- [All Digital Twin Model Demos](#)



ScaleOut Software

Creating a Real-Time Digital Twin
Model for .Net Runtime.

Tracking a Wind Turbine C# Example

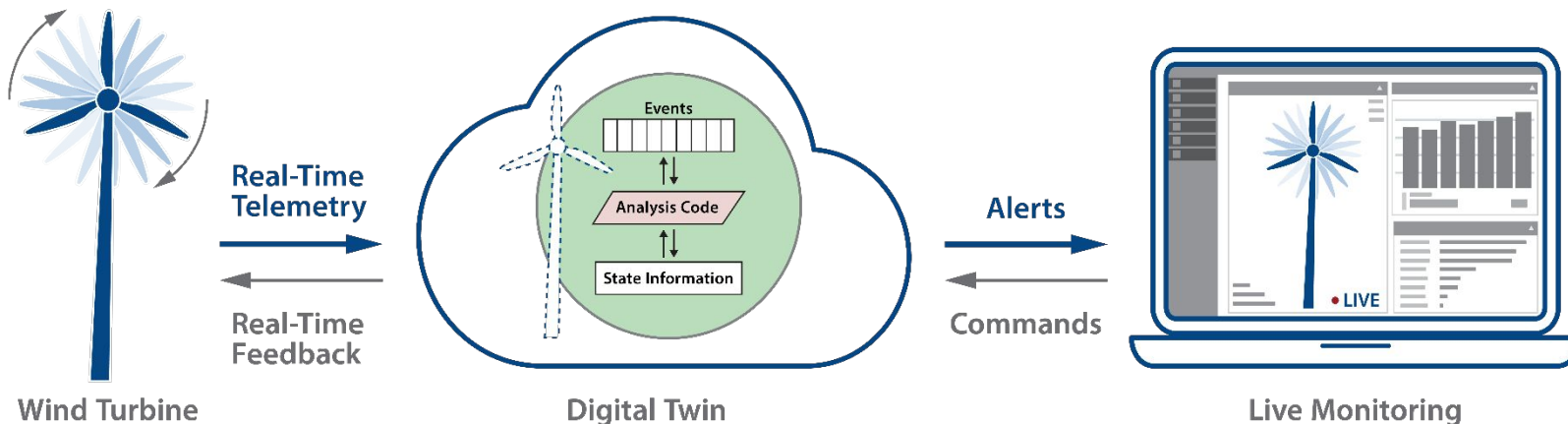
Oleg Shmytov, Senior Director of Engineering
(olegs@scaleoutsoftware.com)

October 27, 2020

- Overview of the wind turbine digital twin model
 - Goals for real-time digital twins
 - Basic principles of creating real-time digital twin models
 - Description of the model code
- Overview of ScaleOut digital twin related NuGet packages
- Demo:
 - Testing the model in a mock environment
- Demo:
 - Deploying and running a wind turbine model in the ScaleOut Digital Twin Cloud Service
 - Processing messages from Azure IoT Hub

Example: Wind Turbine

- Automatically **correlates telemetry** from each data source.
- **Immediately accesses and tracks dynamic, device-specific state** for each wind turbine; for example:
 - Knows turbine's model characteristics, service history and requirements (e.g., time until next service).
 - Tracks recent events (e.g., overtemp/overspeed events).
- **Introspects and alerts in real time** on derived device-specific state information; for example:
 - Tracks frequency and duration of overtemp/overspeed alerts and adjusts feedback based on dynamic analysis.
 - Adjusts time to alerting based on model & characteristics.
 - Adjusts time to alerting if service is upcoming (higher likelihood of unexpected failure).
- **Scales** for 1000s of devices without database & network delays.



Jul 22, 2019



Wind turbine catches fire in Southern Washington

Sample Application with Code (C#)

- Goal: Illustrate use of digital twin to analyze temperature telemetry from a wind turbine.
- Digital twin tracks:
 - *Parameters*: model, pre-maintenance period based on model, max. allowed temperature, max. allowed over-temp duration (normal and pre-maintenance)
 - *Dynamic state*: time to next maintenance, over-temp condition and its duration
- Message processing:
 - Determines onset of and recovery from over-temp condition
 - Alerts at maximum allowed duration based on normal and pre-maintenance conditions
 - Logs incidents for time-windowing analysis



Block Island Wind Farm

Sample State Object (C#)

```
public class WindTurbine : DigitalTwinBase
{
    // physical characteristics:
    public const string DigitalTwinModelType = "windturbine";
    public WindTurbineModel TurbineModel { get; set; } = WindTurbineModel.Model17331;
    public DateTime NextMaintDate { get; set; } = new DateTime().AddMonths(36);
    public const int MaxAllowedTemp = 100; // in Celsius
    public TimeSpan MaxTimeOverTempAllowed = TimeSpan.FromMinutes(10);
    public TimeSpan MaxTimeOverTempAllowedPreMaint = TimeSpan.FromMinutes(2);

    // dynamic state variables:
    public bool TrackingOverTemp { get; set; }
    public DateTime OverTempStartTime { get; set; }
    public int NumberMsgsWithOverTemp { get; set; }

    // list of incidents and alerts:
    public List<Incident> IncidentList { get; } = new List<Incident>();
}
```

Sample Message Processor (Outer Loop)

```
public override ProcessingResult ProcessMessages(ProcessingContext context,
    WindTurbine dt, IEnumerable<DeviceTelemetry> newMessages)
{
    var result = ProcessingResult.NoUpdate;

    // determine if we are in the pre-maintenance period for this wind turbine model:
    var preMaintTimePeriod = _preMaintPeriod[dt.TurbineModel];
    bool isInPreMaintPeriod = ((dt.NextMaintDate
        - DateTime.UtcNow) < preMaintTimePeriod) ? true : false;

    // process incoming messages to look for over-temp condition:
    foreach (var msg in newMessages) {
        // if message reports a high temp indication, track it:
        if (msg.Temp > WindTurbine.MaxAllowedTemp)
            <track over-temp condition>
        else if (dt.TrackingOverTemp)
            <resolve over-temp condition>
    }
    return result;}

```


Track or Resolve Over-Temp Condition

```
// track over-temp condition:
{dt.NumberMsgsWithOverTemp++;

if (!dt.TrackingOverTemp) {
    dt.TrackingOverTemp = true; dt.OverTempStartTime = DateTime.UtcNow;
    <add a notification to the incident list> }

(TimeSpan) duration = DateTime.UtcNow - dt.OverTempStartTime;

// if we have exceeded the max allowed duration for an over-temp, send an alert:
if (duration > dt.MaxTimeOverTempAllowed ||
    (isInPreMaintPeriod && duration > dt.MaxTimeOverTempAllowedPreMaint)) {
    var alert = new Alert(); <fill out the alert message>;
    context.SendToDataSource(Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(alert)))
;
<add a notification to the incident list> }}

// resolve the condition and reset our state:
{dt.TrackingOverTemp = false; dt.NumberMsgsWithOverTemp = 0;
<add a notification to the incident list> }
```

DEMOS

Overview of .NET API library for building real-time digital twin models:

https://static.scaleoutsoftware.com/docs/digital_twin_user_guide/software_toolkit/dt_builder/dotnet_api/dotnet.html

.NET API Reference for the ScaleOut Digital Twin Builder™:

<https://static.scaleoutsoftware.com/docs/DigitalTwinDotNetAPI/html/11a0da96-8c30-4c07-b8a8-f1cc4a4d2080.htm>

Scaleout.Streaming.DigitalTwin.Core NuGet package:

<https://www.nuget.org/packages/Scaleout.Streaming.DigitalTwin.Core/>

Source code for .Net sample applications used in the presentation:

<https://github.com/scaleoutsoftware/DigitalTwinDemos/tree/main/DotNet>

Questions?

Feel free sending me your feedback or questions on **olegs@scaleoutsoftware.com**



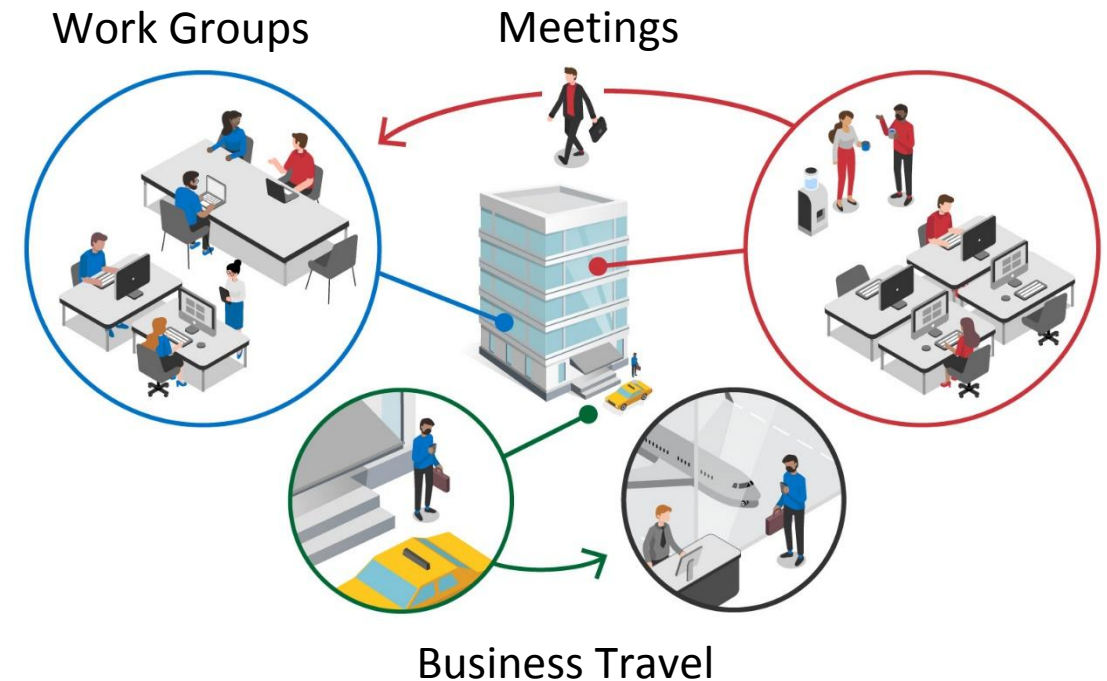
ScaleOut Software

Corporate Contact Tracing Using Real-Time Digital Twins Demonstration

October 27, 2020

Olivier Tritschler, oliviert@scaleoutsoftware.com

- Goals:
 - Demonstrate a technique that could help companies get employees back to work safely.
 - Demonstrate the use of real-time digital twins for timely notifications, improved situational awareness, and fast development.
- Limitations of public contact tracing:
 - Need for time-consuming, manual effort
 - Privacy concerns and lack of cooperation
- Advantages for companies:
 - Limited, known population
 - Known clusters and interactions
 - Ability to implement policies and procedures (fewer privacy concerns)
 - Ability to quickly react to strategic information




Demonstration Application (1)

- Consists of a mobile application and cloud-based tracking service.
- Mobile app for employees:
 - Notifies service of new and recurring contacts outside of work group.
 - Notifies service of community contacts during business travel (flights, taxis, restaurants).
 - Notifies service of a positive test for COVID-19.
 - Receives alert of possible exposure and need for quarantine.

EXAMPLE USE CASE: ECONOS CONTACT TRACING APP

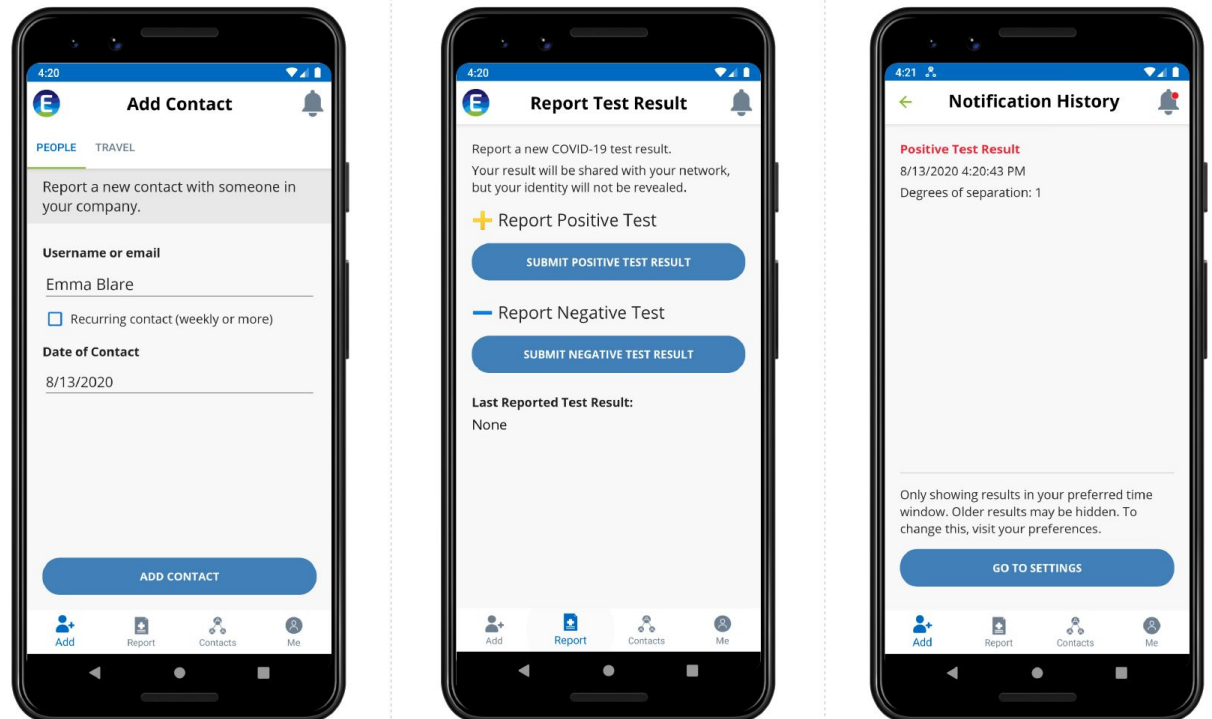
Econos, a fictional multinational energy company, requires its 30,000 employees to use a contact tracing app built on ScaleOut Real-Time Digital Twin technology.



Lisa adds **Emma** as a contact after an in-person meeting

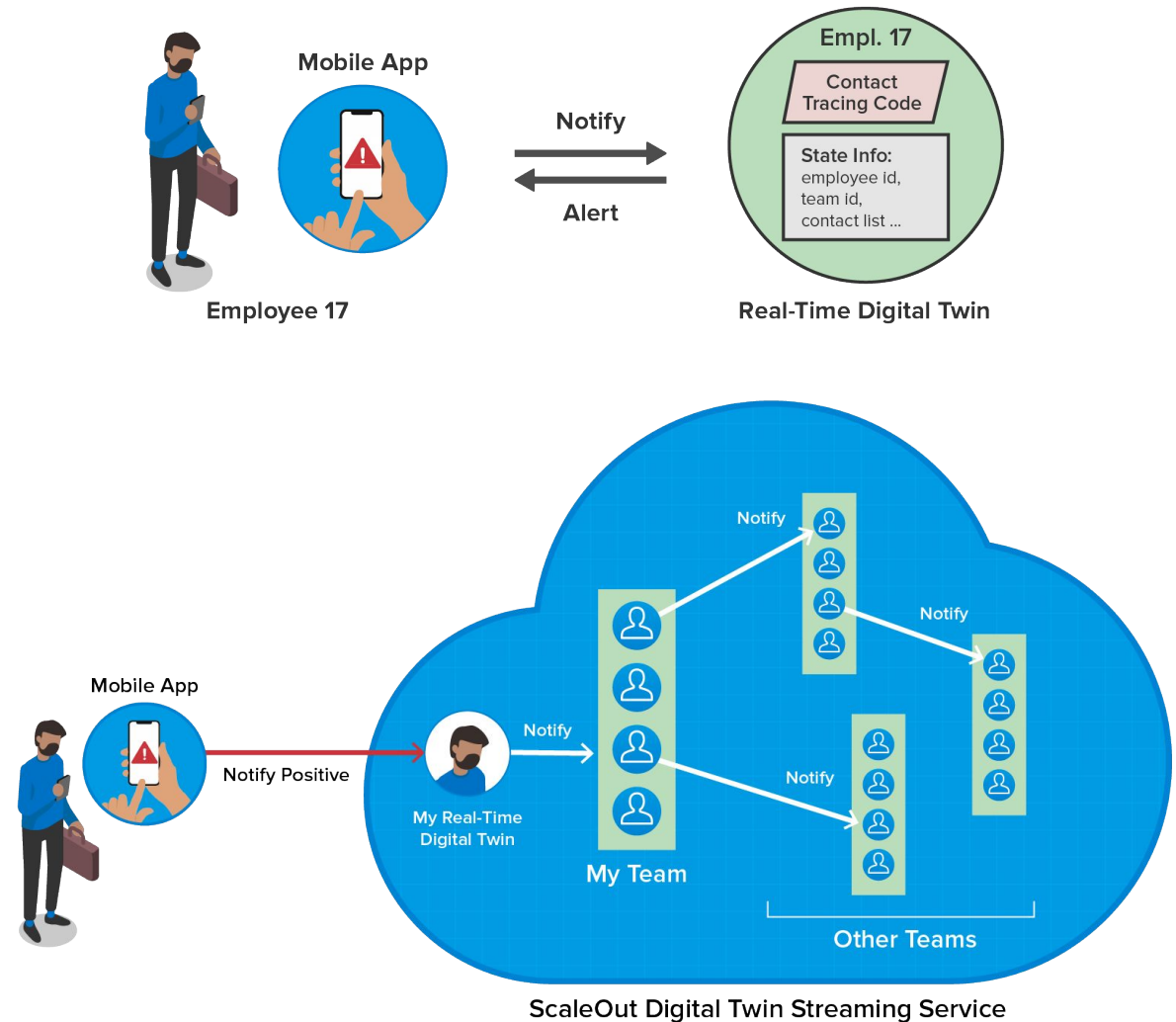
Lisa tests positive for COVID and anonymously reports her test result

Emma immediately gets notified of a possible exposure



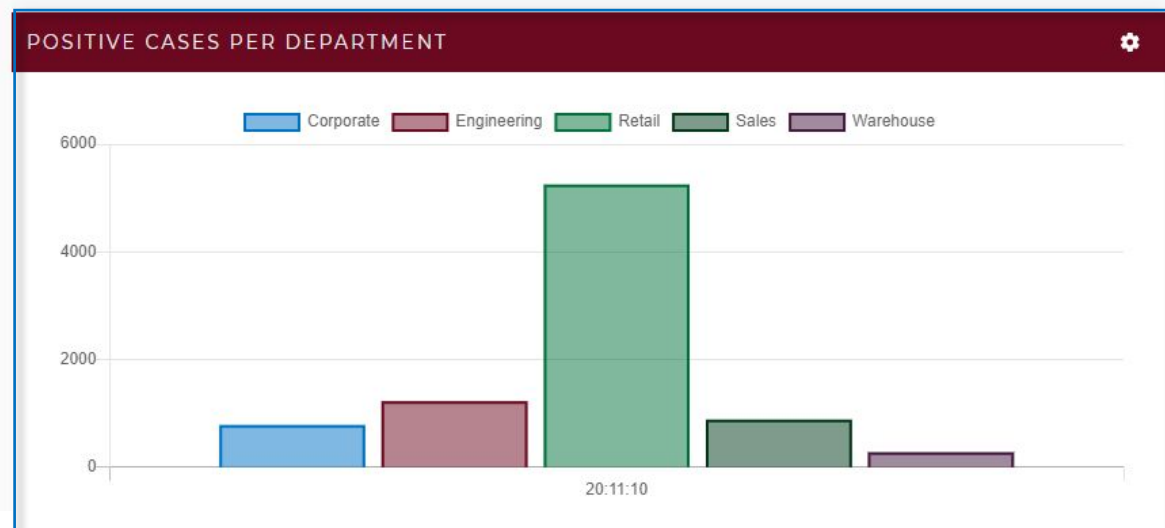
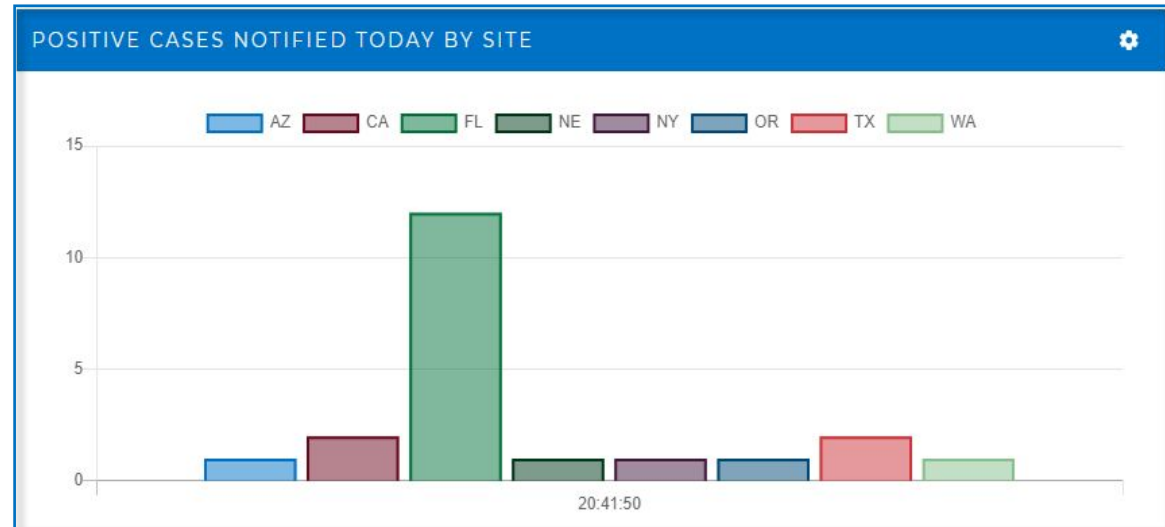
Demonstration Application (2)

- Cloud service tracks employees, creates notifications, and provides real-time aggregate statistics.
- A real-time digital twin instance tracks each employee:
 - Keeps list of contacts.
 - Signals other digital twins when employees tests positive.
 - Digital twins traverse network of contacts within seconds.
 - Each signaled twin alerts its employee.
 - Twins maintain statistics for aggregate analysis.



Aggregate Statistics


- Cloud service combines statistics from all real-time digital twins every few seconds.
- Enables managers to immediately spot clusters of COVID exposures and take action.
- Enables managers to assess high risk areas e.g., (department types, locations) and quickly implement new policies.



Econos: A Simulated Company

- Load generator simulates a company to demonstrate the application's capabilities.
- Works with live mobile app.
- Simulated company profile:
 - 30K employees
 - 5 department types
 - Locations in 8 states
 - 20-55 employees per work group (1,366 groups)
- Load generator:
 - Simulates interactions.
 - Simulates employee notifications of COVID-19 exposure.
 - Runs faster than real-time for demonstration purposes.

Contact Tracing Load Generator

**ScaleOut Software**

Simulation time
One week of simulation takes minute(s) to complete. Simulation date: Tuesday, October 6, 2020 Start Simulation

Probabilistic model for new contacts
Average number of new contacts per simulated hour Next contact is in the same city %
Next contact is in the same department % Next contact is in a different city in the same state %
Next contact is in a different state %

Probabilistic model for new positive tests

State	Infection rate per 100K people/day		State's current stats
Arizona	<input type="text" value="3.9"/>	Spike	New case every 86 real-time seconds.
California	<input type="text" value="0.9"/>	Spike	New case every 194 real-time seconds.
Florida	<input type="text" value="0.9"/>	Spike	New case every 372 real-time seconds.
Nebraska	<input type="text" value="0.9"/>	Spike	New case every 374 real-time seconds.
New York	<input type="text" value="0.9"/>	Spike	New case every 220 real-time seconds.
Oregon	<input type="text" value="0.9"/>	Spike	New case every 255 real-time seconds.
Texas	<input type="text" value="0.9"/>	Spike	New case every 372 real-time seconds.
Washington	<input type="text" value="0.9"/>	Spike	New case every 138 real-time seconds.

Real-Time Digital Twin messages statistics
Signal messages sent per second: 0
Contact messages sent per second: 0

Programming Exercise

Natural Gas Smart Meter Digital Twin Model ScaleOut Software

Natural gas typically consists of methane (or ethane), propane and other elements. This gas is an extremely fast-acting, toxic substance. The maximum allowable short-term exposure level is 50 ppm for 15 minutes



Programming Exercise:

1. Create a digital twin model to process telemetry from a gas sensor for tracking gas concentration and alerting when maximum exposure has been reached.
2. Create messages which report the id, current ppm, and timestamp for a gas sensor.
3. Set an alert flag when either the gas level exceeds 50 ppm for more than 15 minutes or the level spikes to 200 ppm.
4. (Bonus task): in addition to setting the flag add the code for sending a pipe shutdown control message back to the meter device using the `SendToDataSource` method.

Real-time digital twins are:

- **Powerful:** They enable context-tracking and introspection for every data source.
- **Scalable:** They can simultaneously track thousands (or even millions) of data sources.
- **Fast:** They can respond to messages from data sources in milliseconds.
- **Strategic:** They enable immediate, real-time aggregate analytics that boost situational awareness.
- **Easy to Use:** They are easy to develop in Java, C#, or JavaScript (and a rules engine is coming soon).

The ScaleOut Digital Twin Streaming Service provides an integrated, cloud-based platform for hosting real-time digital twins and visualizing aggregate analytics.



ScaleOut Digital Twin
Streaming Service™

**Thank you for joining us today.
We hope you enjoyed the session!**

To learn more, visit us at:

www.scaleoutsoftware.com

Documentation is posted at:

<https://www.scaleoutsoftware.com/scaleout-support/documentation>

The slides and demos are posted at:

<https://github.com/scaleoutsoftware/DigitalTwinDemos>



www.scaleoutsoftware.com