Enhancing Intelligent Real-Time Monitoring with Streaming Analytics Using Real-Time Digital Twins

RTInsights.com

ScaleOut Software

CHAPTER 1

An Architecture for Intelligent Real-Time Processing

The wealth of data available from smart sensors and internet of things (IoT) devices creates an exciting new opportunity to perform intelligent real-time monitoring. The insights derived from this monitoring can be used to dramatically improve responsiveness and situational awareness in many application areas including fraud detection and prevention, fleet management, security, disaster recovery, and health tracking.

The challenge is how to make sense of such vast amounts of data in real time and quickly determine the appropriate response in rapidly evolving situations. If that could be accomplished, such capabilities would provide valuable insights, enabling a switch from a reactive stance to a proactive strategy. Unfortunately, deriving these insights in real time from many data sources is computationally challenging. Traditional techniques, such as stream-processing pipelines combined with batch analytics (or similar examples of the Lambda architecture), defer most streaming analytics to batch processing instead of performing this in real time. This dramatically limits both the introspection on streaming data in real time and the quality of the response.

To support the requirements of intelligent real-time monitoring, what's needed is a data-processing architecture designed to process a very large number of incoming telemetry streams in real time while also providing continuous, aggregate analysis. The derived intelligence can then be used to detect important patterns, respond in real time, and maintain situational awareness. Recent advances in in-memory computing are creating new capabilities that make this possible.

Enter Real-Time Digital Twins

A breakthrough new approach to intelligent real-time monitoring that addresses these challenges and avoids the limitations of the Lambda architecture is the concept of **real-time digital twins**. The term "digital twin" has had different meanings over time. It was originally created for use in predictive maintenance and product life cycle management (PLM) of devices and was based on information derived from digital replicas of physical assets. Real-time digital twins are similar to digital twins in PLM in that they track the dynamic state of physical assets—but that's where the similarity ends.

Real-time digital twins offer unique advantages for intelligent, real-time monitoring, especially for thousands of telemetry streams, such as from car or truck fleets, intrusion sensors, or other IoT applications. By deriving and maintaining dynamic, device-specific information for all data sources and benefiting from automatic event correlation, real-time digital twins can track these telemetry streams, introspect on dynamic changes, and respond in milliseconds. In addition, real-time digital twins provide a basis for aggregate analysis that can identify developing trends in seconds from thousands of devices to detect patterns and enhance overall situational awareness.

Real-time digital twins offer unique advantages for intelligent, real-time monitoring, especially for thousands of telemetry streams, such as from car or truck fleets, intrusion sensors, or other IoT applications.

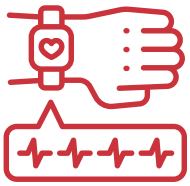
Using real-time digital twins to combine intelligent streaming analytics with aggregate analysis offers several benefits in a wide-ranging set of real-world applications, including:

Disaster recovery: Dealing with rapidly changing large forest fires, first responders typically know that points A, B, and C are on fire. That information is not enough to act. What's needed is immediate analysis of all data sources that provides an understanding of the dynamics of the situation. What direction is the fire moving? How fast is it moving? These insights can enhance the ability of first responders to deploy assets for evacuating people and fighting the fire. This information can be provided by using real-time digital twins to analyze telemetry coming from sensors distributed throughout a forested region combined with real-time aggregate analysis to identify real-time trends.

Cyber security: Most of today's critical infrastructure, such as power grids, water supplies, and refineries, are managed by countless interconnected and highly vulnerable computer systems. Unauthorized intrusions can originate simultaneously at multiple locations and quickly spread across the network to create a widespread outage. When an attack occurs, it is vital to quickly identify the sources of new threats and isolate the affected subsystems, thereby minimizing the attack's overall impact. Real-time digital twins can serve a vital role in tracking and reporting abnormal network activity (such as denial of service attacks) from every computer in the network, and real-time aggregate analysis can pinpoint and then isolate affected subnets within a few seconds. This also allows operations managers to immediately assess the scope of an attack and take action to both limit its reach and maintain ongoing operations.

Medical freezers: The failure of refrigeration units can result in costly and life-threatening situations. For example, a hospital might lose vital organs, pharmaceuticals, or critical blood supplies. Real-time digital twins can receive telemetry from sensors on each refrigeration unit reporting on fast-changing data such as compressor statistics (operating temperature, oil purity, and oil viscosity), refrigerator temperature, current contents load, door open/close events, and more. These digital twins can track and analyze this data in the context of non-telemetry information, such as location, model type, and maintenance history, to enhance the real-time analysis of each unit. Alerts can provide warning of an imminent failure and make use of aggregate analysis to identify widespread outages and suggest appropriate alternate storage locations.

Fleet tracking: Intelligent streaming analysis of a trucking fleet or rental car company's vehicles using real-time digital twins can improve situational awareness for the entire fleet. For example, it can be used to detect regional issues affecting several vehicles, such as weather delays and closed highways, and enable dispatchers to react within seconds. A strategic response in these situations could lower costs and avoid schedule delays. Additionally, these real-time digital twins can track individual vehicles using specific contextual information, such as the intended route, the driver's profile, and the vehicle's maintenance history; these twins can then alert dispatchers when problems (for example, a lost driver or impending engine failure) are detected.



Health monitoring: Real-time digital twins can add significant value in tracking wearable healthcare devices, such as smart watches that track heart rate for a large population of runners or for a high-intensity exercise program. These digital twins can analyze heart rate telemetry and other parameters generated by each smart watch and provide important feedback to its wearer. By maintaining contextual knowledge of the wearer's age, medical history, current medications, type and duration of exercise, and other parameters, each digital twin can intelligently analyze incoming telemetry and detect abnormal conditions. The results of this real-time analysis can be used to immediately alert the participant or medical personnel if it indicates an imminent threat to health. In addition, the results of ongoing, aggregate analysis can be used to notify a participant if their session history over time suggests that a heart rate trend deviates significantly from that of the overall population.

The following chapter explains how the digital twin model evolved from its traditional usage in product life cycle management for a new use in stream processing with the name "real-time digital twins." It also shows how this concept enables much deeper introspection on telemetry in live systems than previously possible with traditional streaming pipelines. Chapter 3 explains how in-memory computing technology has created a breakthrough for hosting thousands or even millions of real-time digital twins while delivering fast event processing, scalable throughput, real-time aggregate analysis, and integrated high availability.

Chapter 4 describes how the object-oriented design of real-time digital twins enables them to be organized into hierarchies to provide streaming analytics for complex systems that incorporate multiple levels of subsystems, from devices to high-level controllers. The last chapter leverages the object-oriented concept one step further, explaining how real-time digital twins can transparently migrate to the edge—where devices are located—and minimize event-processing latency without the need to redesign real-time analytics software.

Real-time digital twins offer a powerful new approach to streaming analytics that should dramatically increase the quality of both real-time introspection and of overall situational awareness. By harnessing the power of in-memory computing, real-time digital twins are poised to fundamentally change what we think is possible in managing complex, live systems.

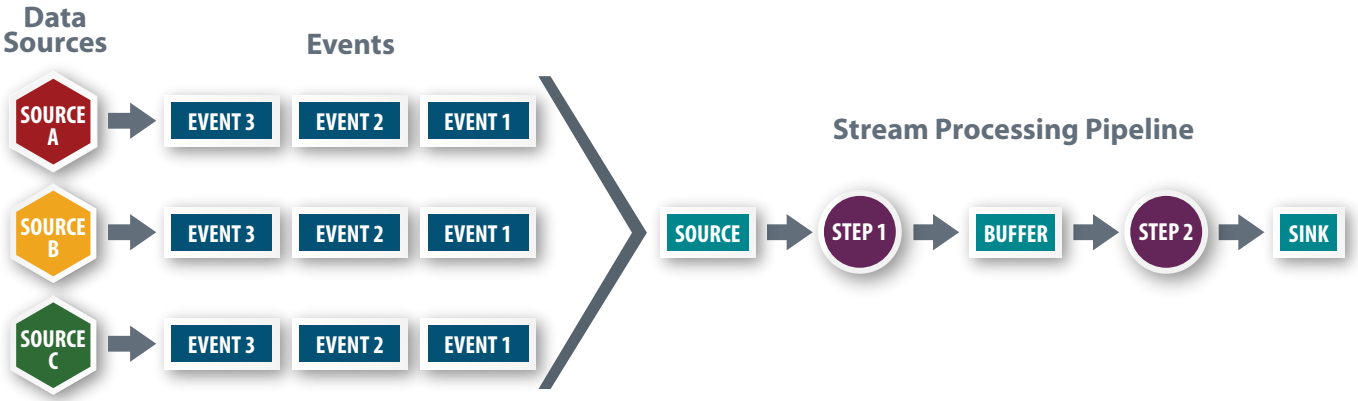
CHAPTER 2

The Real-Time Digital Twin: A Foundational Concept for Stateful Stream Processing

Traditional stream-processing and complex event-processing systems (such as Apache Storm or Software AG's Apama) have focused on extracting interesting patterns from incoming data using stateless applications. While these applications maintain state information about the data stream itself, they don't generally make use of dynamic information about the data sources or their context.

For example, a stateless IoT application which attempts to detect whether data from a temperature sensor is predicting the failure of the medical freezer would look at patterns in the sequence of temperature changes. The investigation might identify sudden spikes or a continuously upward trend, without regard to the freezer's usage or service history.

Stateless stream-processing applications are typically constructed as a stream-processing pipeline which simultaneously processes events from many data sources (see diagram).

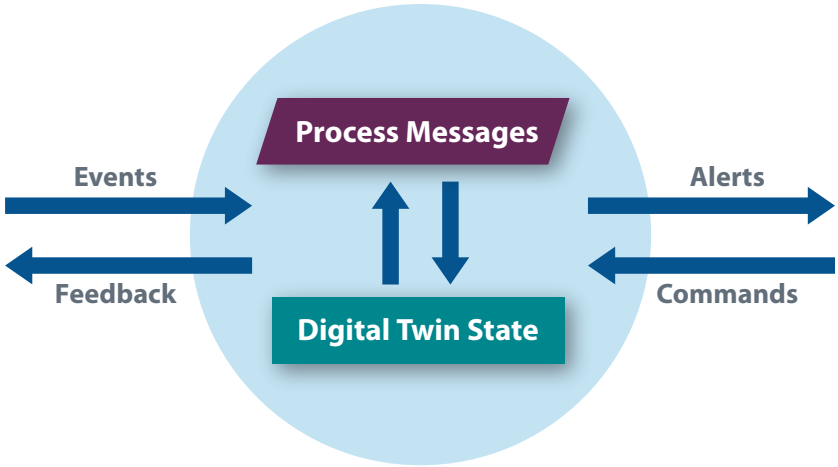


More recent stream-processing platforms, such as Apache Flink, have incorporated stateful stream processing into their architectures by holding state information using key-value stores or databases that the application can make use of to enhance its analysis. However, they do not offer a specific semantic model that applications can leverage to organize and track useful state information and thereby deepen their ability to analyze data streams.

What's missing in these platforms can be found in a "real-time digital twin" model. While the term "digital twin" was coined by Dr. Michael Grieves (U. Michigan) in 2002 for use in product life cycle management, more recently it has been popularized for IoT by Gartner. This model offers key insights into how state data can be organized within stream-processing applications for maximum effectiveness. Using a modified version of digital twins, called *real-time* digital twins, applications can implement a stateful model of the individual physical data sources that generate event streams and maintain separate state information for each data source.

The real-time digital twin model provides an intuitive approach to organizing state data, and, by shifting the focus of analysis from the event stream to the dynamic behavior of the data sources, it potentially enables much deeper introspection than previously possible. With the real-time digital twin model, an application can conveniently track all relevant information about the evolving state of each physical data source. It can then analyze incoming events in this rich context to provide high-quality insights, alerting, and feedback. For example, real-time digital twins of medical freezers could track detailed facts about each freezer's specific model, its service history, environmental conditions, and usage patterns to help analyze telemetry and make more informed predictions about possible impending failures. In fact, using telemetry plus other stored data, derived state can be calculated and tracked by the real-time digital twin to improve introspection.

Beyond providing a powerful semantic model for stateful stream processing, real-time digital twins also offer advantages for software engineering because they can take advantage of well-understood object-oriented programming techniques. A real-time digital twin can be implemented as a data class that encapsulates both state data (including a time-ordered event collection) and a method for analyzing incoming messages and updating state data. Analytics methods can range from simple sequential code to machine-learning algorithms or rules engines (see diagram). These methods also can reach out to databases to access and update historical data sets.



For each physical data source, an instance of a real-time digital twin model is created by the stream processing system to receive and analyze events. It is the responsibility of the system to correlate data from a given data source for delivery to each instance of a digital twin. In many applications, a stream-processing system may host thousands or more real-time digital twins to handle the workload from its data sources.

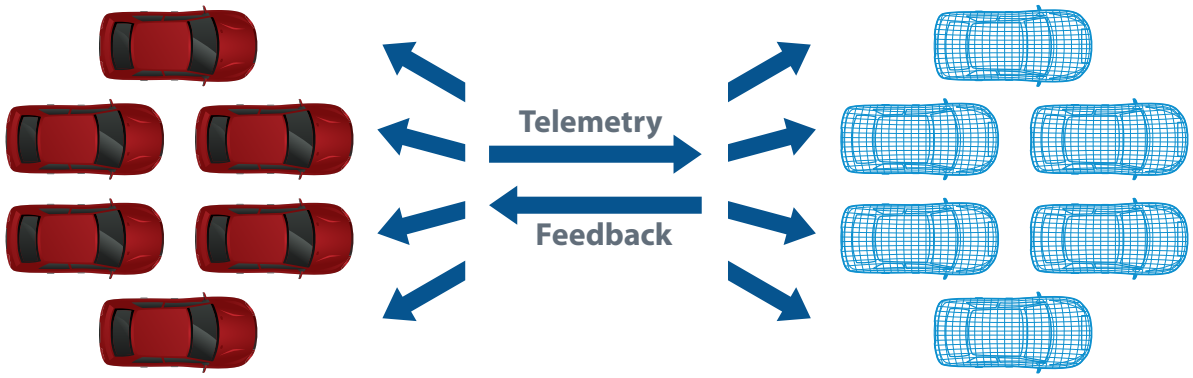
One last point to consider is the granularity of a real-time digital twin. Does it encompass a model of a single sensor or that of a subsystem comprising multiple sensors? As with object-oriented programming in general, the answer is in the hands of the application developer, who must make choices about which data (and event streams) are logically related and need to be encapsulated in a single entity for analysis to meet the application's goals. For example, a real-time digital twin for a medical refrigerator might track telemetry from multiple sensors (temperature, voltage, door position, etc.) so that it can integrate this information to best predict an imminent failure.

The real-time digital twin model provides a powerful organizational tool that focuses on the state of data sources instead of just the telemetry within event streams. With this additional context, it magnifies the developer's ability to implement deep introspection and represents a new way of thinking about stateful stream processing.

CHAPTER 3
The Power of an In-Memory Data Grid for Hosting Real-Time Digital Twins

The secret to the power of the real-time digital twin model—and its excellent fit for execution on an in-memory computing platform—is its focus on organizing and analyzing incoming telemetry and dynamic state data for the specific device or other data source to which it corresponds. All event messages from each data source are correlated and delivered in time-order to an instance of a real-time digital twin, which tracks both dynamic state information and historical knowledge of that data source. With this approach, a stream processing application has a rich context for analyzing event messages and determining what actions need to be taken in real time.

For example, consider an application that tracks a rental car fleet to look for drivers who are lost or driving recklessly. The application can use the real-time digital twin model to correlate real-time telemetry (e.g., location, speed) for each car in the fleet and combine that with data about the driver's contract, safety record with the rental car company, and possibly driving history. Instead of just examining the latest incoming events, the application now has much more information instantly available to judge when and whether to signal an alert (see diagram).



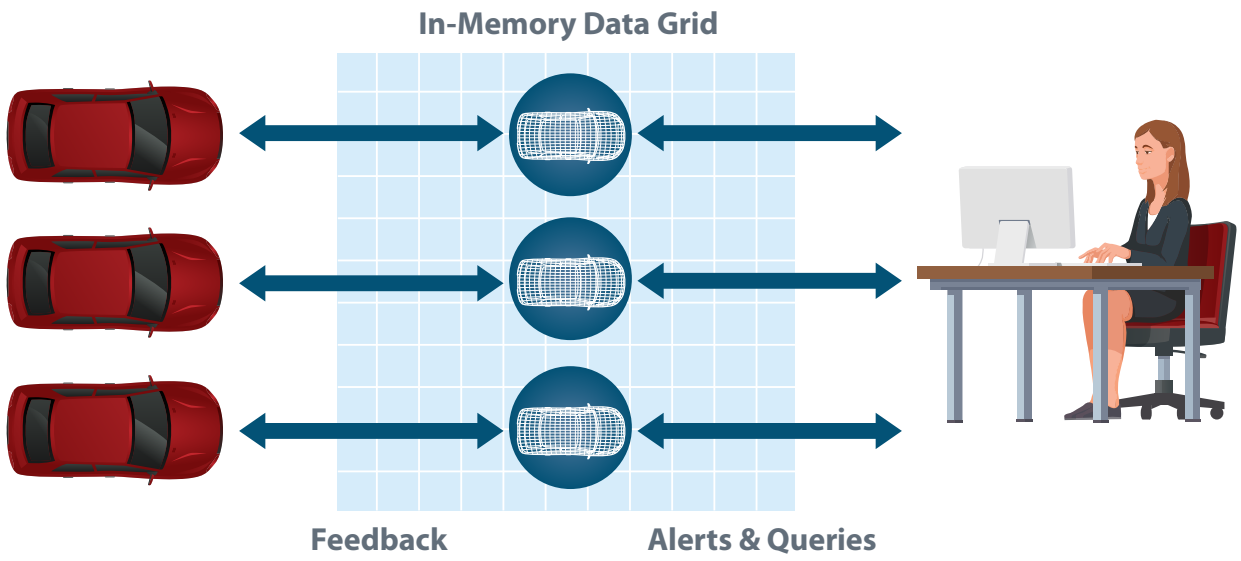
When implementing a stateful stream-processing application using the real-time digital twin model, its object-oriented approach offers an intuitive means to manage data and event processing. For each type of data source, the developer can create a data type (object "class") that describes real-time state data and a message-processing method to be executed when a new event message arrives. Instances of this data type then can be created for each unique data source as its real-time digital twin for use in stream processing.

The use of an object-oriented approach enables an in-memory data grid (IMDG) with integrated in-memory computing (e.g., ScaleOut StreamServer) to serve as an excellent stream-processing platform for real-time digital twin models. An IMDG implements a software-based, key-value store of serialized objects that spans a cluster of commodity servers (or cloud instances). Its architecture provides cost-effective scalability and high availability while hiding the complexity of distributed in-memory storage from the applications that use them. It also can take full advantage of a cluster's computing power to run application code *within* the IMDG—where the data lives—to maximize performance and avoid network bottlenecks.

An IMDG stores instances of real-time digital twin models as objects within the grid, and it processes incoming events by running the model's message-processing method in the grid's software compute engine. This mapping of real-time digital twins to an IMDG automatically correlates incoming event messages by data source and delivers messages to the grid object corresponding to the event's data source. An IMDG can store many thousands of real-time digital twin instances and scale storage capacity and event-processing throughput as needed by transparently adding servers to the IMDG. Event messages typically can be processed in 1-3 milliseconds, ensuring low latency in responding to data sources.

As part of event processing, real-time digital twins can create alerts for human attention and feedback directed at the corresponding data sources. In addition, the collection of real-time digital twin objects stored in the IMDG can be queried or analyzed using data-parallel techniques (e.g., MapReduce) to extract important aggregate patterns and trends. The IMDG's integrated, data-parallel compute engine typically can complete a MapReduce operation in 5-10 seconds, enabling aggregate analysis to be continuously performed in real time instead of deferring it to offline, batch processing. Aggregate results can be fed to dashboards and/or fed back to real-time digital twin models to enhance analysis. This maximizes overall situational awareness for live applications.

For example, the rental car application could alert managers when a driver repeatedly violates criteria specific to the driver's age and driving history. It also could allow a manager to query the details of a specific vehicle to investigate a dynamic situation. Using aggregate analysis, a manager also could immediately determine the status of all cars in every region, for example to analyze the effect of flooding or highway blockages and respond in a timely manner. These data flows are illustrated in the following diagram.



Because they are hosted in memory, real-time digital twin models can react very quickly to incoming events. The IMDG can scale by adding servers to keep event processing times fast, even when the number of instances (objects) and event rates become very large. Although the in-memory state of a real-time digital twin holds the event and state data needed for real-time processing, the application also can reference historical data from external databases as needed to broaden its context. For example, the rental car application could access driving history only when incoming telemetry indicates a need for this information. It also could store past events in a database for archival purposes.

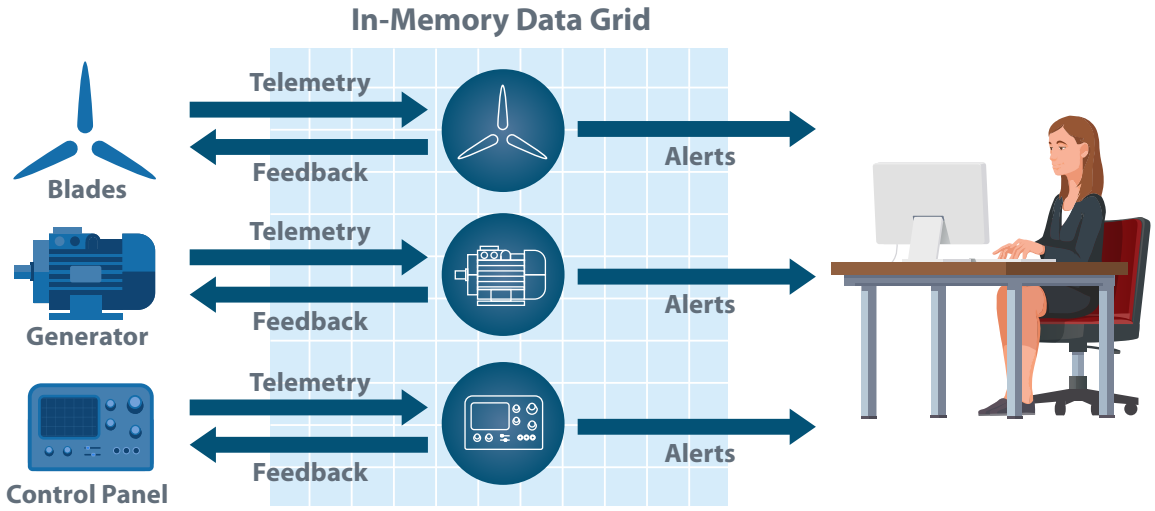
In summary, what makes an IMDG an excellent fit for stateful stream processing is its ability to transparently host both the state information and application code within a fast, highly scalable, in-memory computing platform and then automatically direct incoming events to their respective real-time digital twin instances within the grid for processing. These two key capabilities (correlating events by data sources and analyzing these events within the context of the real-time digital twin's real-time state information) combined with real-time aggregate analysis give developers powerful new tools and new ways to think about stateful stream processing.

CHAPTER 4
The Benefits of Software Architecture: Hierarchical, Real-Time Digital Twins

The key to mainstream adoption of in-memory computing software platforms is *architecture*—the root of a platform's value to applications. The importance of architecture remains as true as ever, although it is often overlooked by application developers, who have deadlines to hit, and by platform developers, who have features and APIs to ship. These priorities tend to push the architecture to the back burner. But careful analysis of application requirements often results in important insights that influence the platform's architecture and lead to new value for *all* applications.

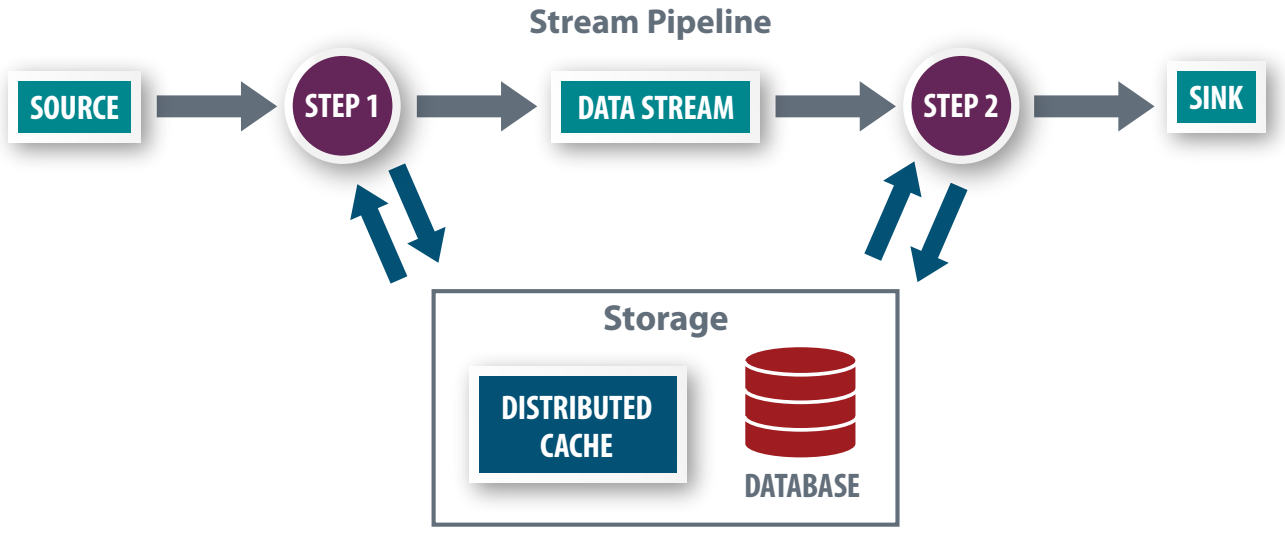
The real-time digital twin model for stateful stream processing is a case in point. This model for building stream-processing applications defines a new software architecture that naturally shifts the application's focus from the event stream to the data sources that are sending events. By correlating incoming events and co-locating all relevant state information for each data source, the real-time digital twin model ensures that a stream-processing application has the context it needs to analyze the data source's dynamic state and generate effective real-time feedback.

For example, if an application is analyzing telemetry from the components of a wind turbine, it can zoom in on the telemetry for each component and combine this with relevant contextual data (such as the component's make, model, and service history) to enhance its ability to predict impending failures. A set of real-time digital twin models can correlate telemetry from three components of a hypothetical wind turbine (blades, generator, and control panel) and deliver this information to associated objects within an in-memory data grid (IMDG). From there, event handlers can analyze the telemetry and generate feedback and alerts (see diagram).



The power of the real-time digital twin model is that this software architecture gives applications new capabilities, simplifies development, and boosts performance. While the model alone does not provide specific APIs for predictive analytics or machine learning, its architecture provides an organizational structure for hosting application-specific algorithms so that they have immediate access to the context they need for deep introspection.

Standard, pipelined models for stream processing, such as Apache Beam, require the application to orchestrate the correlation of events by data source and manage each data source's state information in unstructured attached storage, as shown in the following diagram:

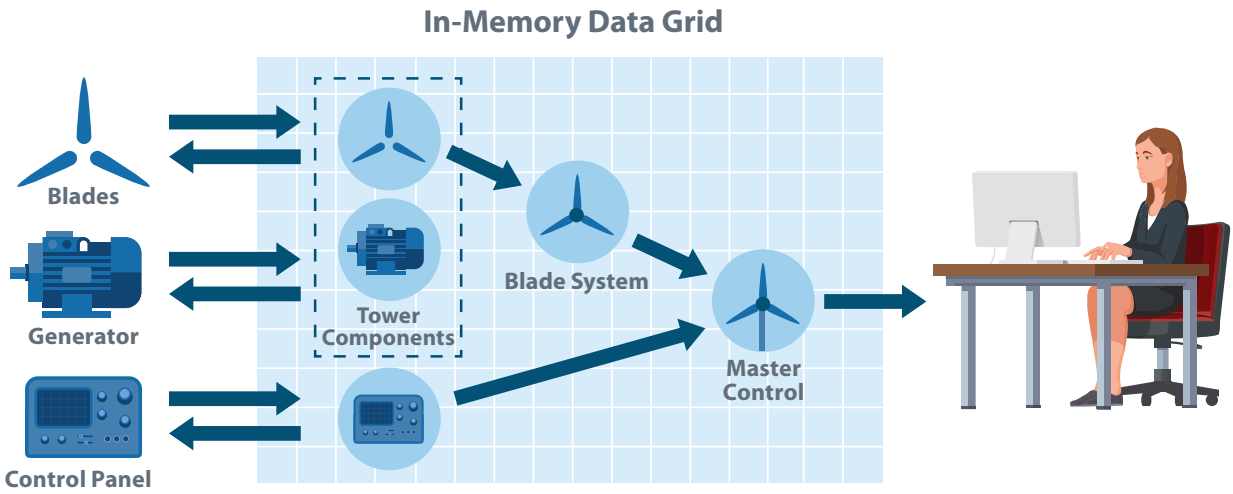


This approach increases complexity and network overhead. When implemented on an IMDG with integrated in-memory computing, the real-time digital twin model provides a new abstraction that both simplifies application design and avoids network accesses to a remote store. As a result, it makes sense to factor its functionality out of application code and migrate it into the stream-processing architecture.

One indicator of a useful software architecture is that it provides unexpected benefits. Beyond just using real-time digital twins to model physical data sources, they can be organized in a hierarchy to implement subsystems operating at successively higher levels of abstraction within a real-time application. Alerts from lower-level real-time digital twins can be delivered as telemetry to higher-level twins.

In the wind turbine example, the blades and generator work together to generate power managed by the control panel. Taking advantage of a hierarchical organization, the real-time digital twins for the blades and generator feed telemetry to a higher-level real-time digital twin model (or blade system) that manages the rotating components within the tower and their common concerns (such as avoiding over-speeds), while not dealing with the detailed issues of managing these two components. Similarly, the real-time digital twin for the blade system and the control panel can feed telemetry to a yet higher-level real-time digital twin model that coordinates the overall wind turbine's operation and generates alerts as necessary.

The following diagram illustrates this hierarchy of digital twins:



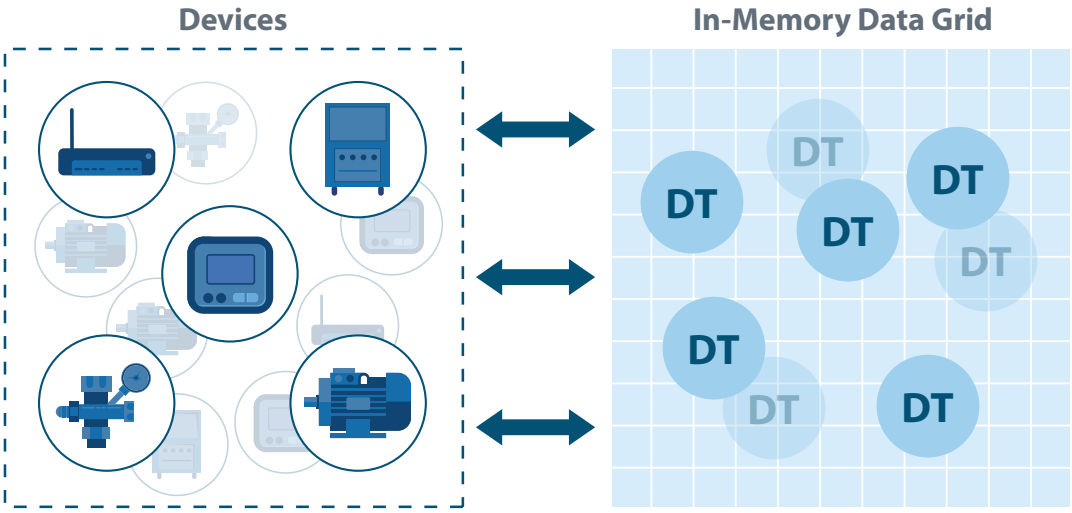
By partitioning the application hierarchically, the code can be modularized and thereby simplified with a clean separation of concerns and well-defined interfaces for testing. In many ways, the real-time digital twin model is just an example of the principle of encapsulation from object-oriented programming applied to the data sources and higher-level controllers within a real-time, stream-processing system. The software architecture of real-time digital twins is so simple that it might be overlooked or trivialized. However, this model significantly simplifies application development and enables the platform to maximize stream processing performance. It thereby offers several benefits for applications, especially when they are hosted on an IMDG with full support for real-time digital twins.

CHAPTER 5
Real-Time Digital Twins Enable Seamless Use of Edge Computing in IoT

Real-time digital twin models are software abstractions that can track the behavior of individual devices in IoT applications. They combine an event-handling function with state information about each device. This state information is used to track evolving device status and help analyze incoming events. It enables deeper introspection on the evolving status of the device than would be possible by just examining the events alone. Also, the use of real-time digital twin models provides automatic correlation of incoming events for each device, thereby simplifying applications.

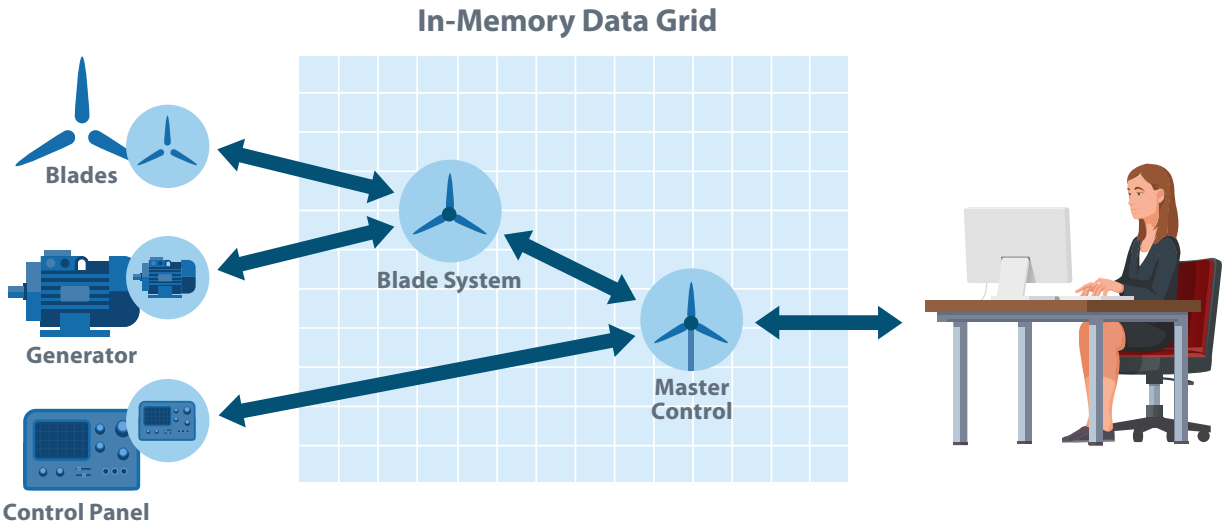
Instances of real-time digital twins can be organized in a hierarchy in which the lowest-level twins track telemetry from individual devices, and higher-level twins implement subsystems that control these devices. Higher-level twins receive events from lower-level twins, and the lowest-level twins receive events from physical devices or other data sources. Likewise, twins at all levels can send messages downwards in the hierarchy for purposes of control—eventually resulting in signals being sent to the devices.

Observing that real-time digital twin models cleanly match the semantics of object-oriented programming, we can straightforwardly implement real-time digital twin instances as memory-based objects that correspond to individual devices or higher-level subsystems. Because real-world IoT applications can track thousands of devices or other entities (e.g., medical patients, e-commerce shoppers), computational power becomes an issue. Distributed, in-memory data grids (IMDGs) with integrated in-memory computing (such as ScaleOut StreamServer) provide a natural platform for hosting these objects and executing their event-handling functions. IMDGs enable transparent performance scaling, which is required to ensure fast event handling, and they typically have built-in high availability (see diagram).



As computing power at the edge inexorably grows, it makes increasing sense to provide enhanced intelligence as close to the data sources as possible. Moving intelligence to the edge minimizes event-handling latency and enables better management of local operations, while still providing strategic analysis and control by remote (cloud-based or on-premises) IoT applications. The challenge is to determine how to partition application logic between the cloud and edge, and more specifically, how to easily migrate functionality to the edge. What is required is a software architecture that enables seamless migration without requiring application code to be reimplemented for execution on edge-based platforms.

The real-time digital twin provides a powerful answer to this challenge. Data and code encapsulation can be used to transparently migrate low-level event-handling and analysis functionality to the edge—where the devices live. Instead of re-implementing application code for use at the edge, the application can simply migrate the lowest-level real-time digital twins to run on edge-based execution platforms without changing their code or messaging protocols. For example, a wind turbine's real-time digital twins can be migrated downwards to the wind turbine itself while keeping the overall real-time digital twin hierarchy intact (see diagram).



The lowest-level real-time digital twins can be hosted at the edge next to their corresponding devices without any changes to the code. Container-based execution can replicate the IMDG's execution environment so that application code is unaware of the migration other than by observing dramatically lower event-handling latency. The higher-level real-time digital twins can continue to run in the cloud or on-premises—wherever the required computing resources are located.

Looking beyond this basic example, the use of the real-time digital twin model does not require that all device-specific functionality migrate to the edge. Consider a more complex application in which a device is represented by both a low-level twin that directly manages the device's operations and a higher-level twin that implements predictive analytics. The predictive analytics might involve using a compute-intensive, machine-learning (ML) algorithm that operates on telemetry received from the low-level twin. Migrating the low-level twin to the edge can improve responsiveness and keep operations running. At the same time, keeping the high-level, "strategic" twin in the cloud ensures computing resources are available for refreshing its predictive analytics algorithm with continuous machine learning.

Transparent migration of event-handling functionality to the edge represents yet another way the real-time digital twin model adds value to stateful, stream-processing applications. The model takes full advantage of object-oriented concepts to simplify application design and create new capabilities. Such new capabilities would be daunting and complex to implement at the application level or with conventional stream-processing platforms.

Using an in-memory data grid (IMDG) to host real-time digital twins offers several important benefits. An IMDG provides both a fast, scalable, highly available, message-processing platform and an object-oriented, in-memory data store for quick access to state information. These features enable a clean mapping to the real-time digital twin model. The use of an IMDG also minimizes network bottlenecks to ensure predictably fast processing—all without complicating the application's structure.

Together, these capabilities make the real-time digital twin model worth a close look when designing the next generation of stream-processing applications for IoT. Real-time digital twins allow businesses to more fully leverage IoT data and respond to changes more quickly and effectively. The derived insights from continuous, real-time analysis of this data provides businesses with a better understanding of the dynamic state of their assets and maximizes situational awareness.

KEY BENEFITS OF AN IMDG

- Automatic event correlation by data source
- Fast message processing (1-3 ms.)
- Continuous, real-time aggregate analysis (5-10 sec.)
- Fast access to state information
- Integrated high availability

RTInsights.com
Accelerate Your Business with Real-Time Insights

RTInsights is an independent, expert-driven web resource for senior business and IT enterprise professionals in vertical industries. We help our readers understand how they can transform their businesses to higher-value outcomes and new business models with IoT real-time analytics. We provide clarity and direction amid the often confusing array of approaches and vendor solutions. We provide our partners with a unique combination of services and deep domain expertise to improve their product marketing, lead generation, and thought leadership activity.

ScaleOut Software

ScaleOut Software develops leading-edge software products for hosting real-time streaming analytics that maximize situational awareness. Its stream-processing platform and cloud service enable the creation of real-time digital twin models of IoT devices that provide immediate feedback, alerting, and integrated, data-parallel analytics. This platform transparently scales to handle millions of data sources and enables an entire population of data sources to be continuously analyzed to quickly identify and respond to broad trends, such multiple intrusions or cascading failures. Real-time digital twins are useful in a wide variety of applications, including intelligent real-time monitoring, IIoT, security, safety, transportation, healthcare, and many others.

Founded in 2003, ScaleOut Software offers a wide range of in-memory computing products that help mission-critical enterprise applications store and analyze live, fast-changing data. Its product portfolio includes ScaleOut StateServer®, an award-winning, linearly scalable, and highly available in-memory data grid for distributed caching, and ScaleOut StreamServer®, a software platform for stateful stream-processing with integrated data-parallel computation on live data. Please visit www.scaleoutsoftware.com for more information on ScaleOut Software's technologies and products.