In-Memory Computing SUMMIT | VIRTUAL EVENT 2020

Integrating Real-Time Stream Processing and Data-Parallel Analytics Using Digital Twins

William Bain, Founder & CEO
ScaleOut Software, Inc.
October 29, 2020

About the Speaker

Dr. William Bain, Founder & CEO of ScaleOut Software:

- Email: wbain@scaleoutsoftware.com
- Ph.D. in Electrical Engineering (Rice University, 1978)
- Career focused on parallel computing – Bell Labs, Intel, Microsoft

ScaleOut Software develops and markets In-Memory Data Grids, software for:

- Scaling application performance with in-memory data storage
- Operational intelligence: analyzing live data in real time with in-memory computing
- 15+ years in the market; 450+ customers, 12,000+ servers

Agenda

- Challenges for Stream Processing for Large Numbers of Data Sources
- Real-Time Digital Twin Software Model
- Target Applications & Examples
- Code Sample
- Using an In-Memory Data Grid (IMDG) to Host Real-Time Digital Twins
- The Role of Digital Twins in Aggregate Analytics
- Implementing Aggregate Analytics Using an IMDG
- Demo

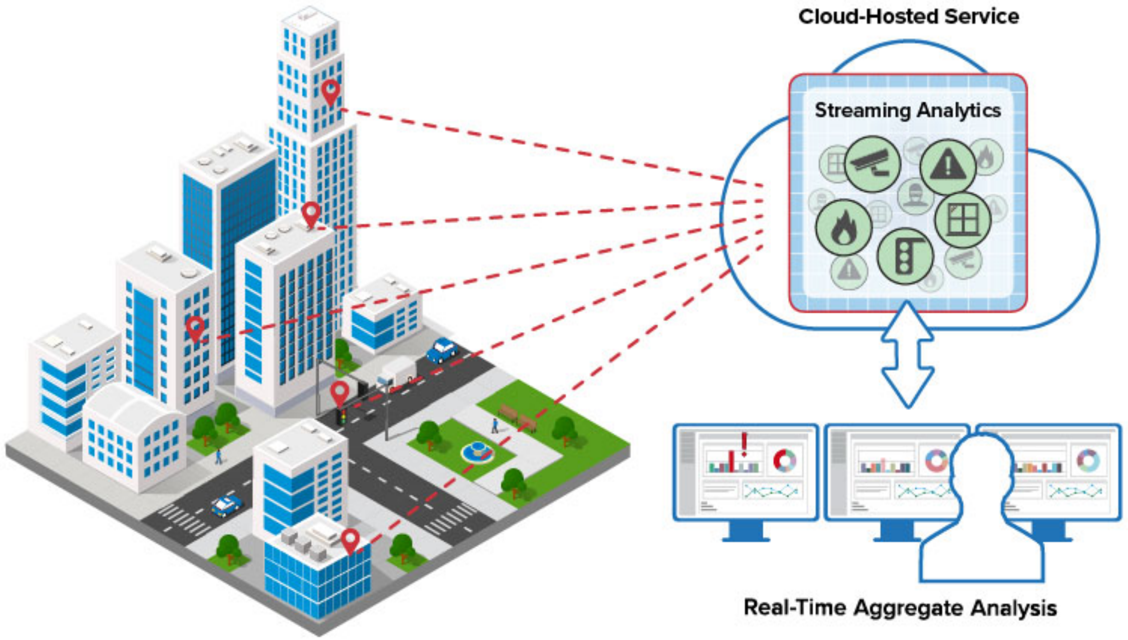
Goals and Challenges

Goals:

- Track the state of many data sources.
- Predict future conditions & emerging issues.
- Respond and alert in real time.
- Maximize situational awareness.

Challenges:

- How to maintain state for each data source?
- How to scale to handle many data sources?
- How to perform aggregate analytics in real time?

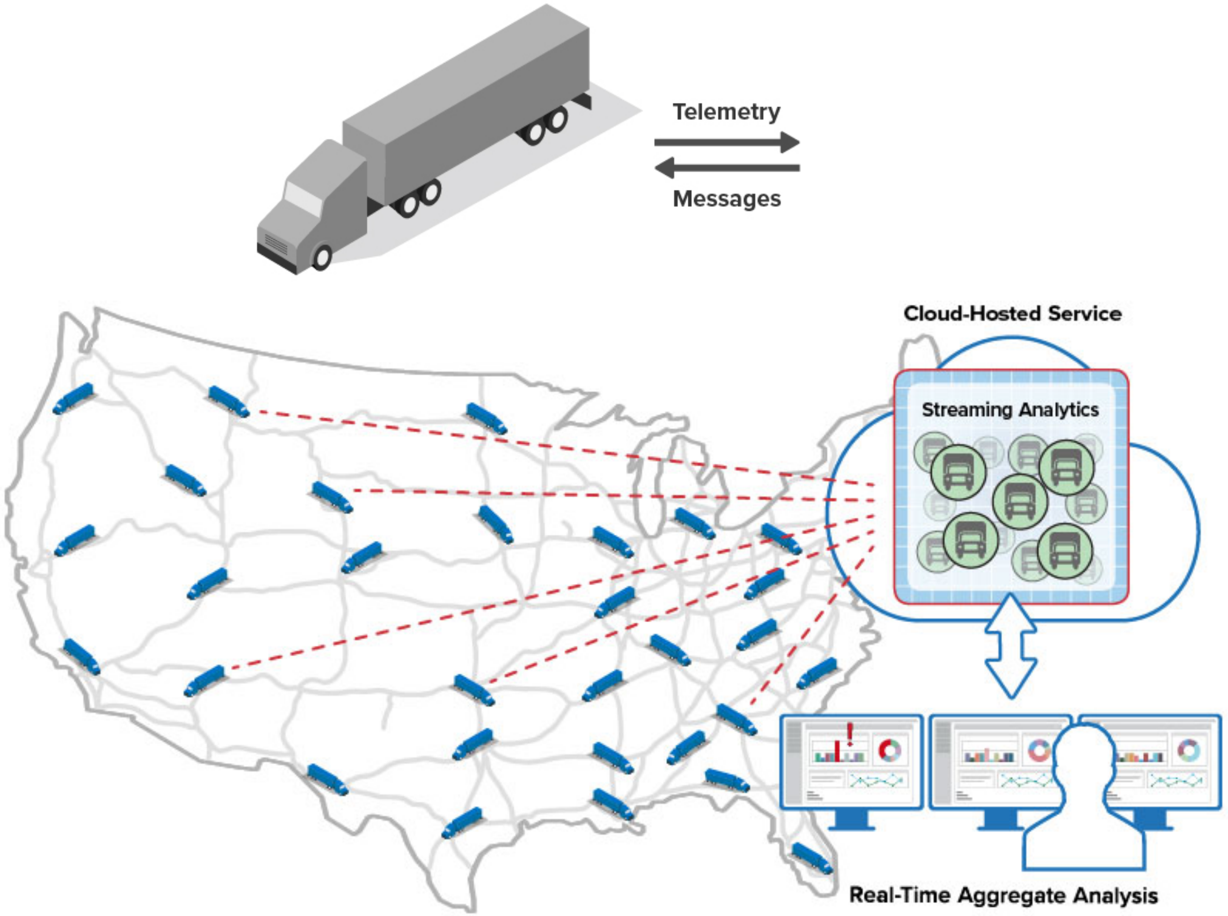


A Smart Cities Application

Example: Fleet Telematics

Track a fleet of trucks

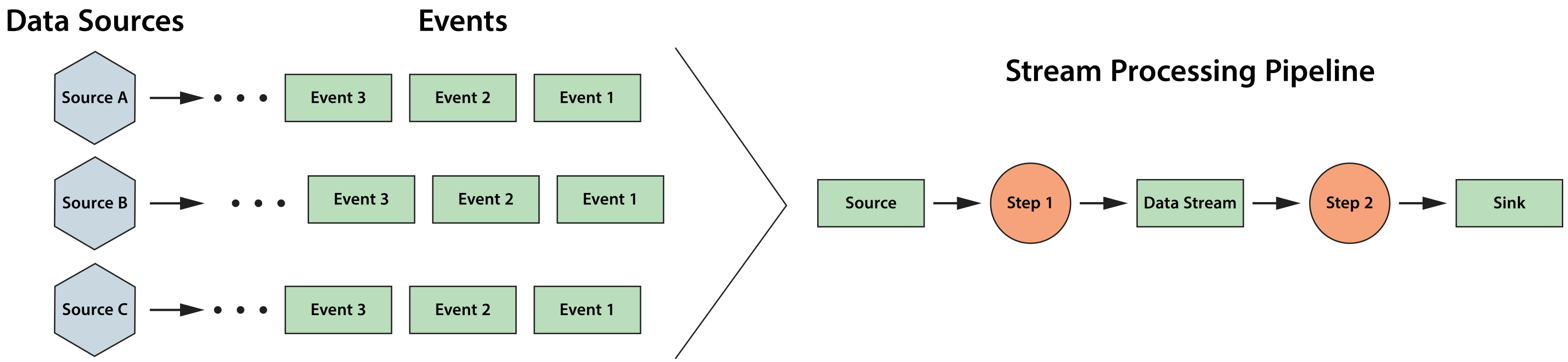
- Trucks have sensors that report to dispatcher every minute.
- Telemetry includes position, speed, engi parameters, cargo parameters.
- Streaming analytics determines:
  - Emerging issues with vehicle and cargo
  - Delays vs. route and schedule
  - Lost, fatigued, or timed-out drivers
  - Overall fleet performance & issues



Challenges for Streaming Analytics

Challenges for tracking large numbers of data sources:

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

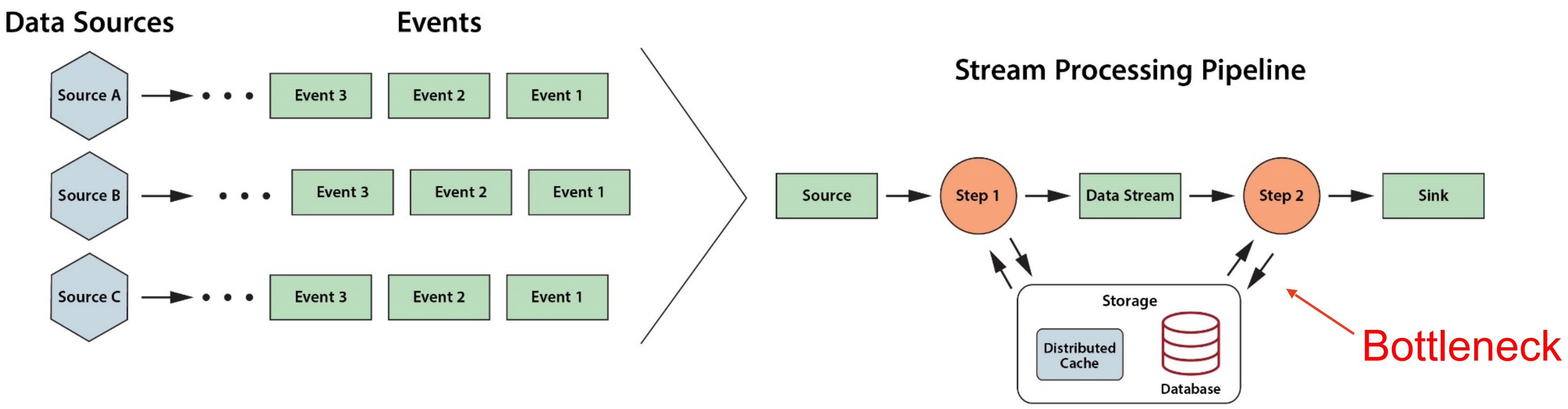


- Creates **complexity challenges**: correlating messages by data source, partitioning work.
- Creates **performance challenges**: achieving scalable speedup, avoiding network overhead.

Managing Contextual Data

How to track dynamic state information for each data source?

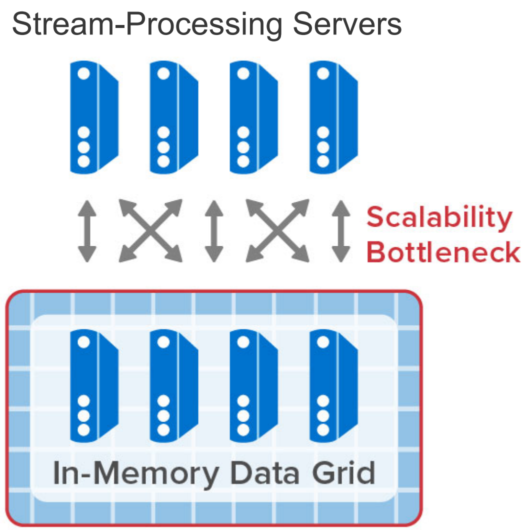
- Pipelined streaming platforms typically do not maintain integrated, in-memory contextual information for each data source.
- This can create **network bottlenecks** when accessing external stores:

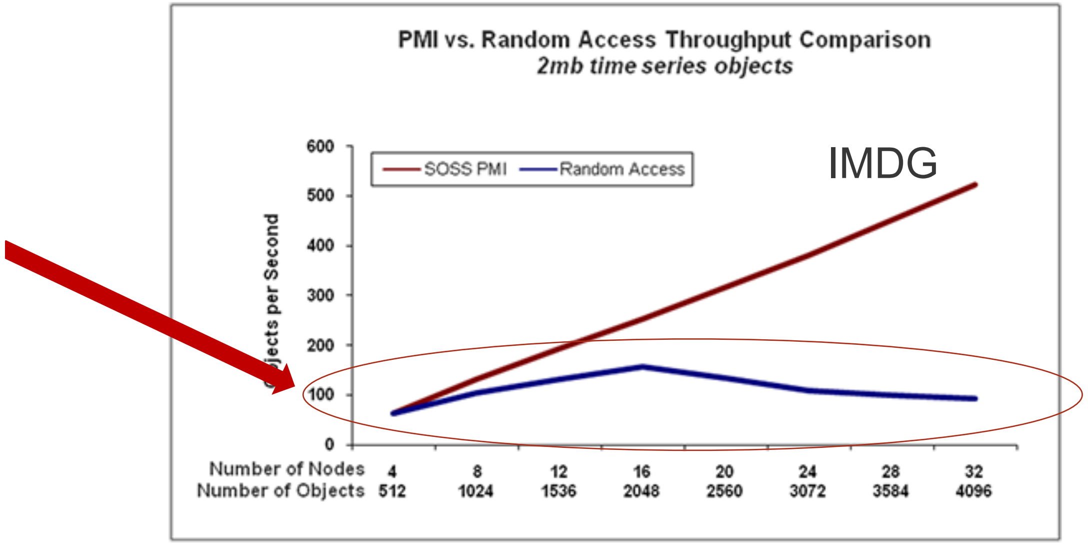


The Impact of Network Bottlenecks

Network bottlenecks can limit throughput scaling:

- Accessing contextual data from an external store creates delays in stream processing.



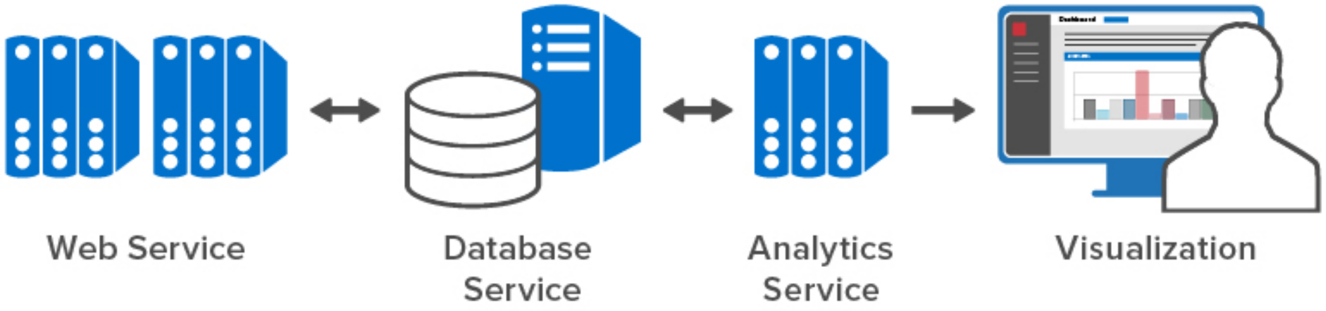


Example of the Effect of Network Bottlenecks

Ad Hoc Techniques

Applications often track data sources by combining cloud services:

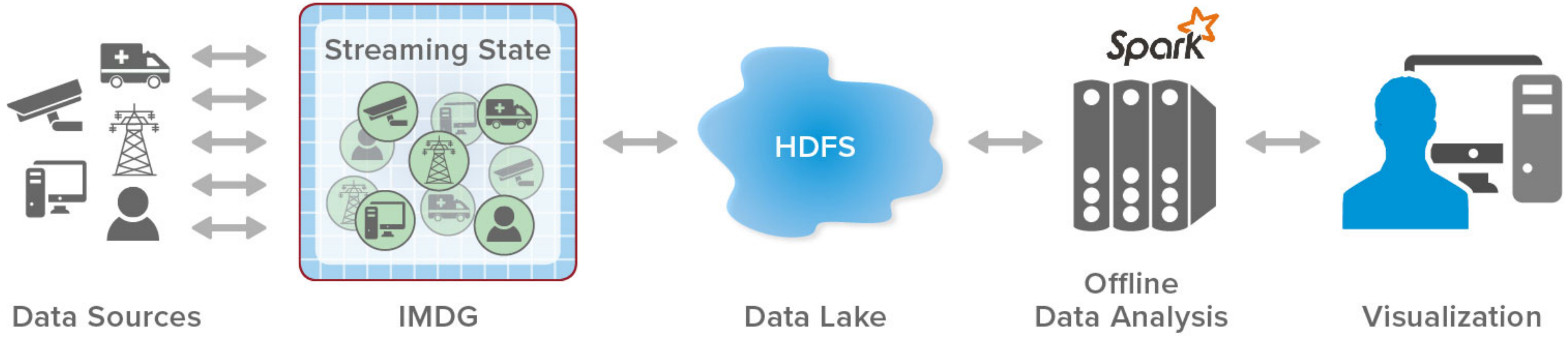
- Ad hoc techniques typically use a front-end web service, application servers, database and/or blob stores, offline analytics, and visualization.
- This requires several skills, can introduce bottlenecks, and uses **offline aggregate analytics**.



Limitations of Batch Aggregate Analytics

Applications need to maximize overall situational awareness:

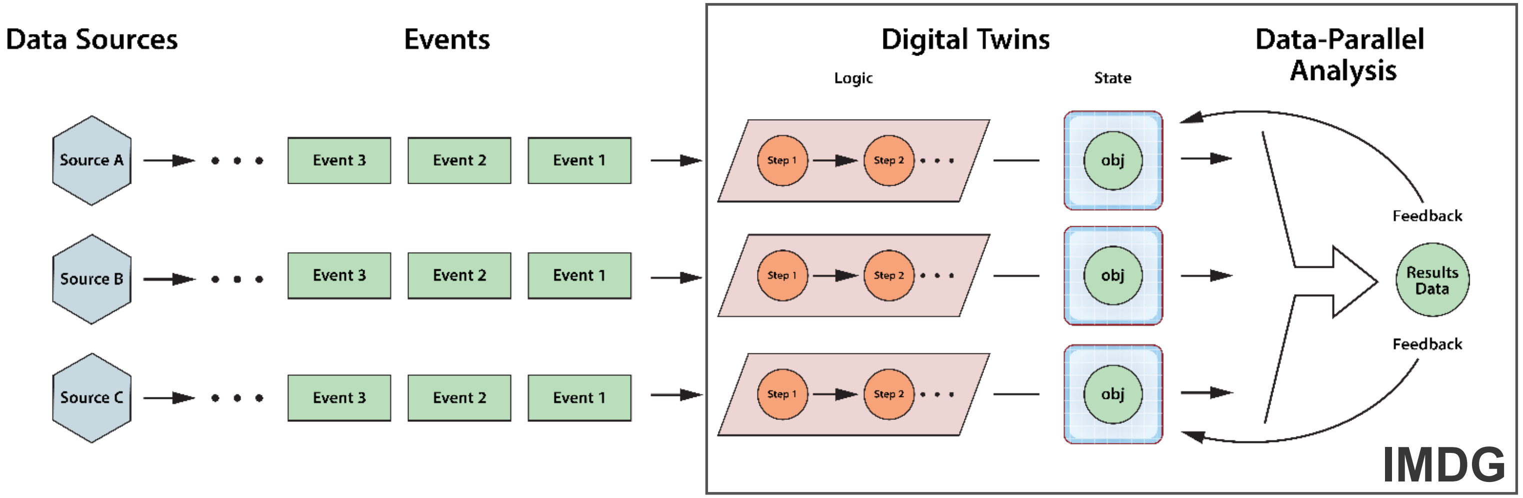
- This **requires immediately aggregating contextual state** information about data sources.
- Pushing data to a data lake for offline processing by a “big data” platform (e.g., Spark) creates delays (minutes or hours) that impact situational awareness.



Real-Time Digital Twins

A new software technique for tracking large numbers of data sources:

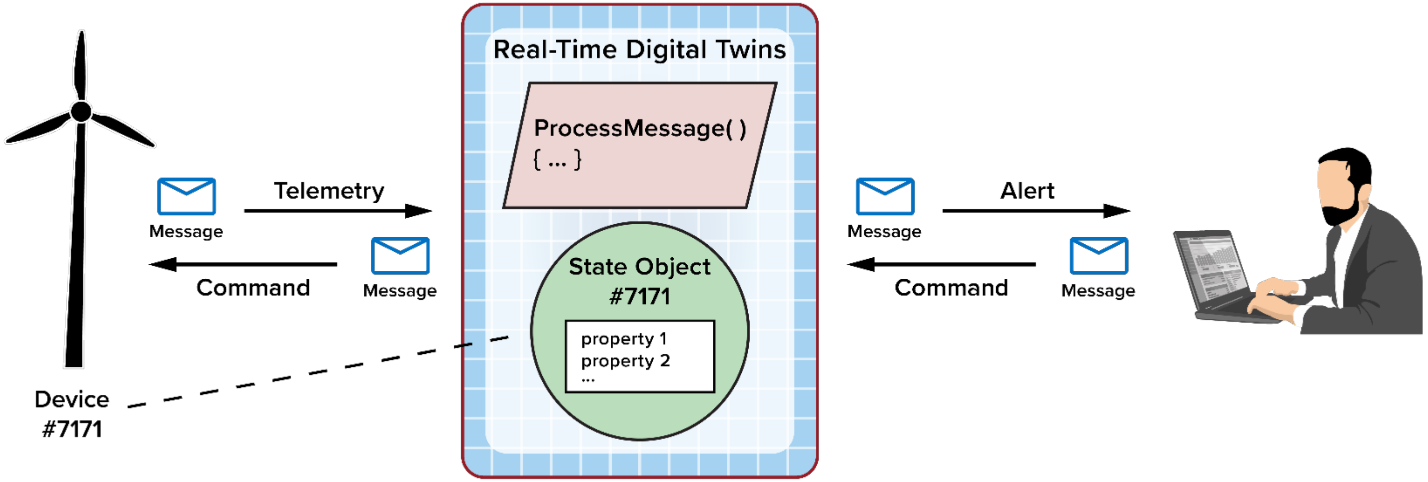
- Focus on **state tracking** by maintaining dynamic state information for each data source.
- Automatically **correlate telemetry** from each device or data source for processing.
- Provide a software **framework** for hosting application logic (e.g., rules, ML).



Anatomy of a Real-Time Digital Twin

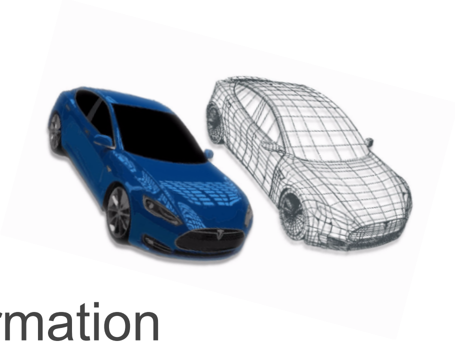
A real-time digital twin model describes how to process incoming messages from a specific type of data source:

- **State object** defines properties of the data source to be tracked (one instance per data source).
- **ProcessMessages** method implements application-specific code that analyzes incoming messages using the state object and then responds, commands, or alerts as necessary.



History and Other Uses of Digital Twins

Digital twins are used in multiple contexts:



- Originally described by Michael Grieves for product lifecycle management.
- Also used to describe device parameters or hierarchical relationships:
  - **AWS device shadow**: cloud-based repository for per-device state information
  - **Azure IoT device twin**: JSON document that stores per-device state information
  - **Azure digital twin**: spatial graph of spaces, devices, and people for modeling relationships in context
- **Real-time digital twins** focus on streaming analytics.

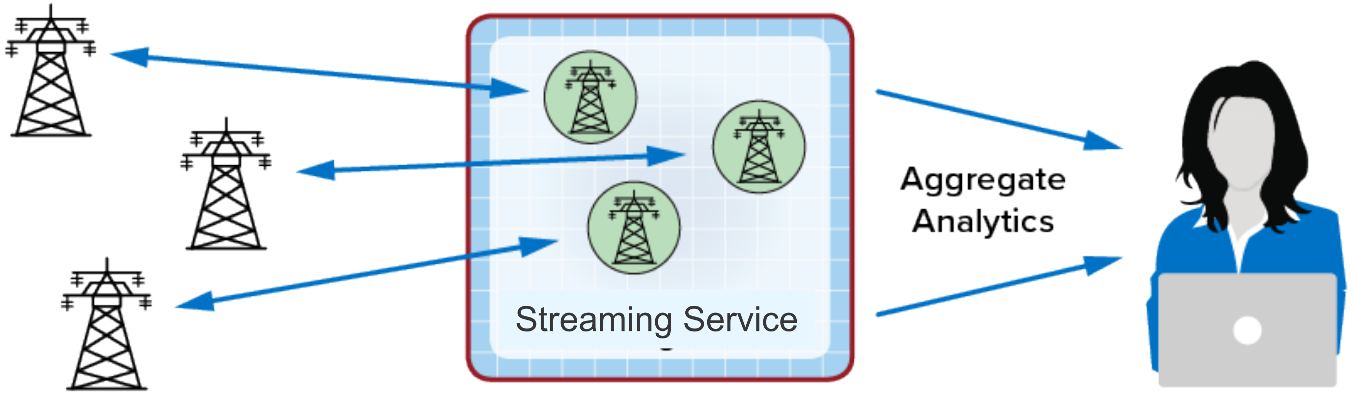
> A digital twin may be used for simulation, as a kind of prototype to understand expected behavior, existing before there is a physical twin. It can also capture real-world behavior so that, for example, analytics and learning can be performed. …

Definition from the Digital Twin Consortium

Advantages of Real-Time Digital Twins

Real-time digital twins enable tracking of large number of data sources:

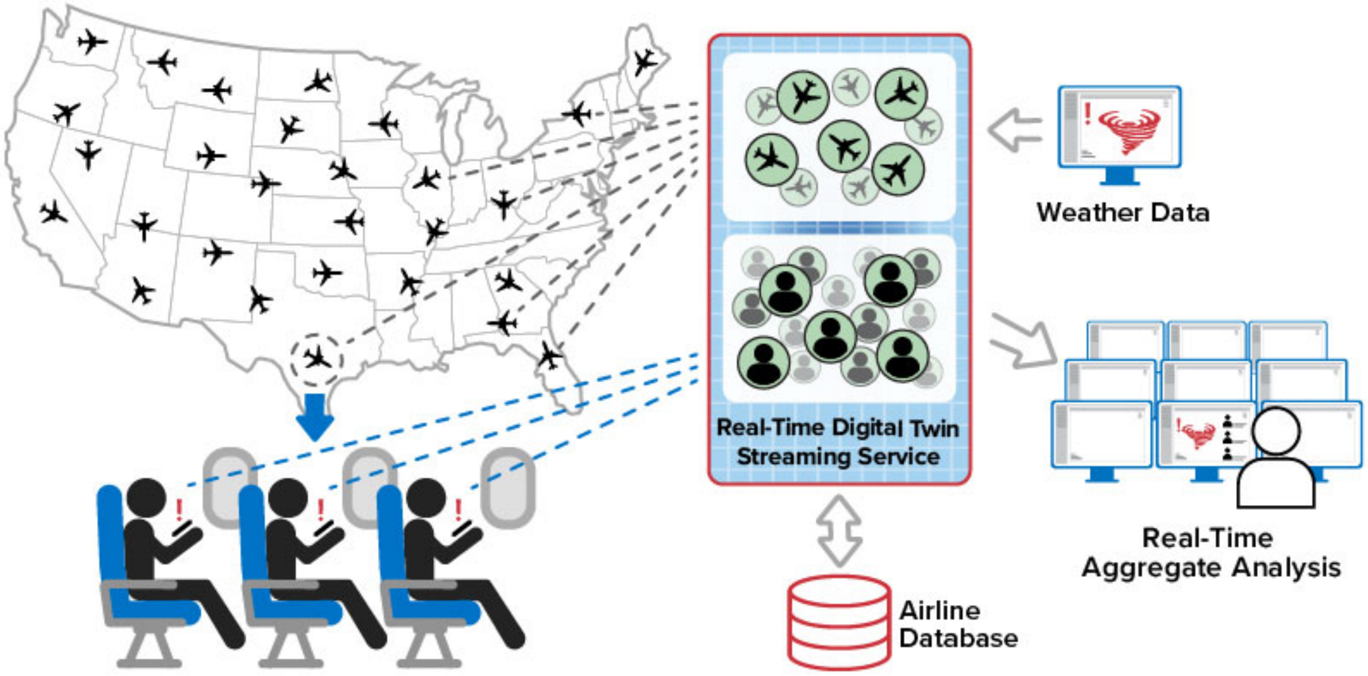
- Provide a **simple,** flexible software model for encapsulating application code (e.g., predictive analytics, rules, ML). They avoid the need for message correlation by data source.
- Enable **deep introspection** with state tracking for each data source.
- Enable **fast** responses, commands, and alerts by avoiding network delays to access state data.
- Transparently **scale** message handling using an IMDG.
- Provide a basis for real-time **aggregate analytics**.



Many Target Applications

Real-time digital twins assist in "real time intelligent monitoring" to maximize situational awareness for live systems:

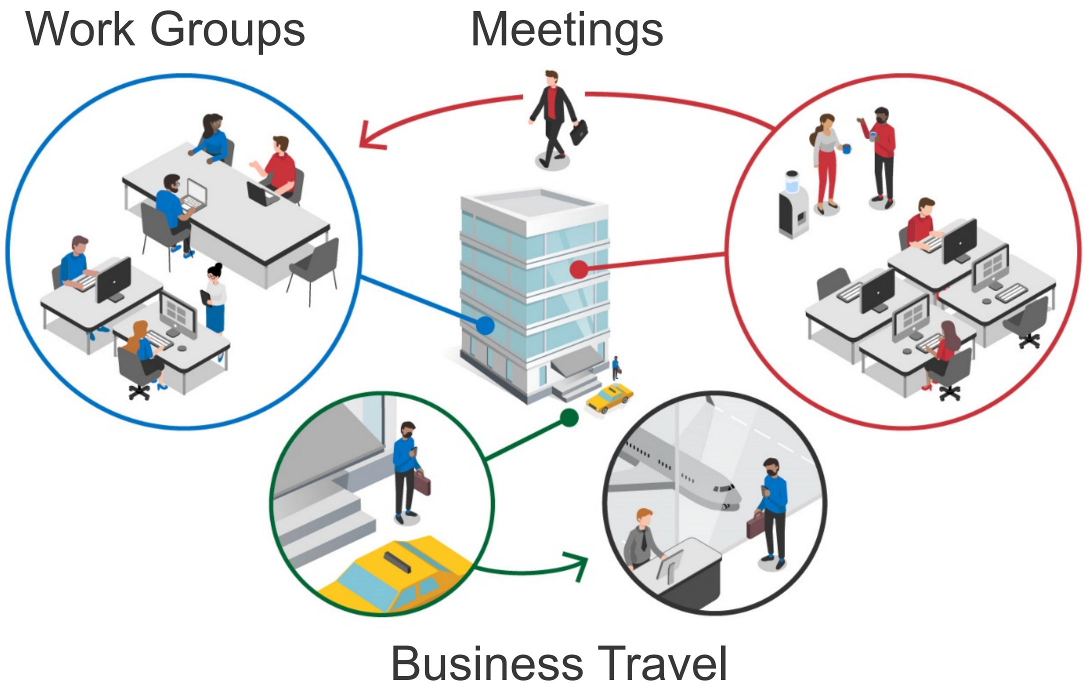
- IoT and smart cities
- Fleet telematics & logistics
- Contact tracing
- Security & disaster recovery
- Health-device tracking
- Ecommerce recommendations
- Financial services (e.g., fraud detection)



Example: Contact Tracing for Companies

Real-time digital twins can track employee contacts within a company to quickly notify employees exposed to COVID-19:

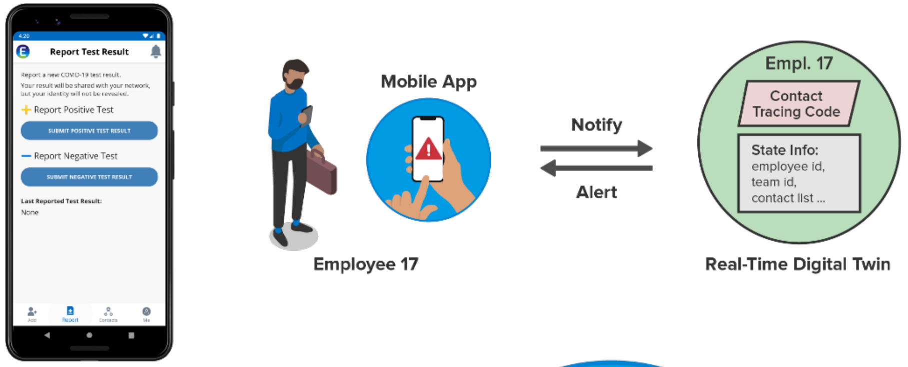
- Public contact tracing has numerous obstacles to adoption (e.g., privacy).
- Companies need fast notifications.
- They can take advantage of:
  - Known clusters and interactions
  - Ability to implement policies
  - Ability to quickly react to evolving situations and control exposures.

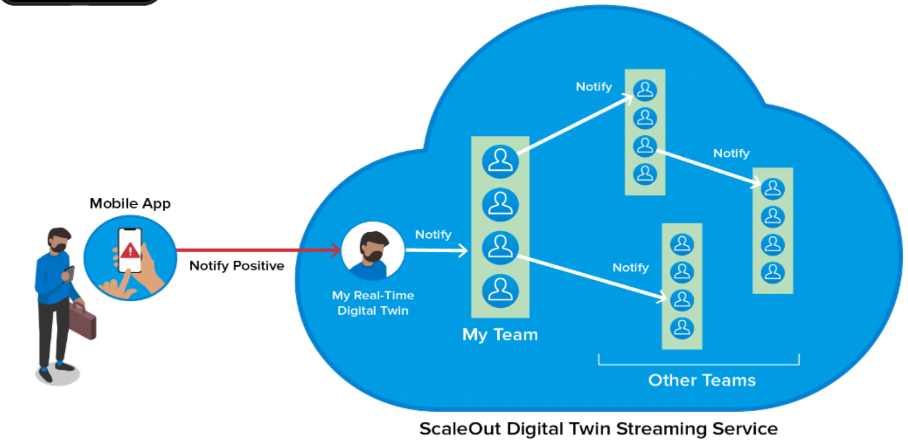


Using Digital Twins for Contact Tracing

A real-time digital twin instance can track each employee:

- Keeps list of contacts notified by employee using mobile app.
- Signals other digital twins when employee notifies that tests positive.
- Digital twins traverse network of contacts within milliseconds.
- Each signaled digital twin alerts its employee using mobile app.
- Digital twins maintain statistics for aggregate analysis.

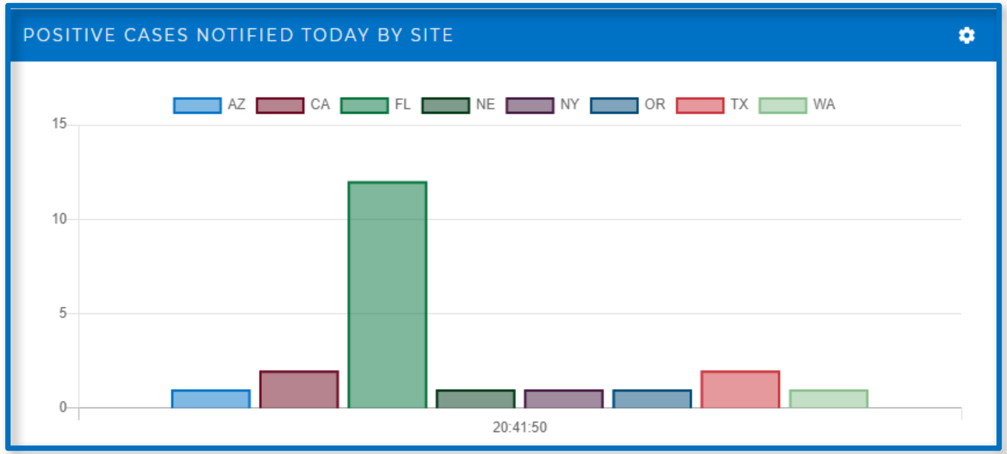


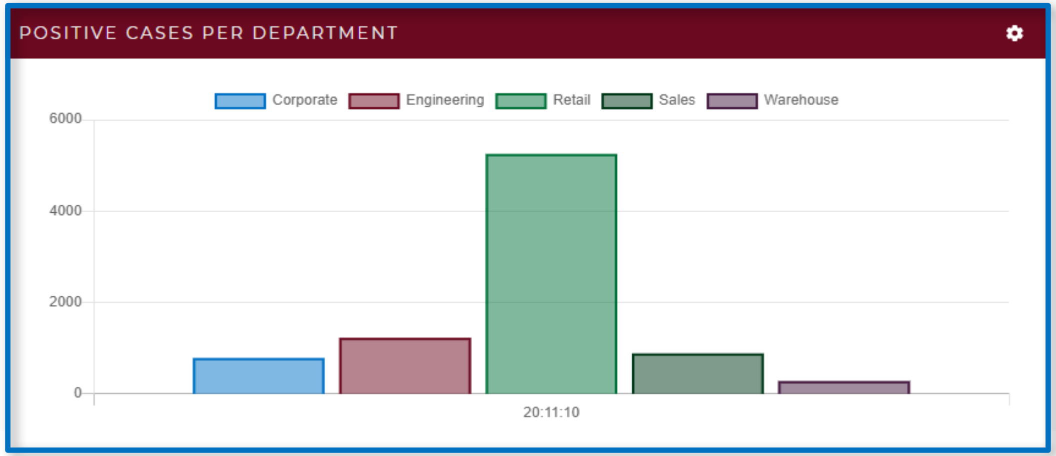


Benefits of Aggregate Analytics

Aggregate analytics help identify "micro-clusters" of COVID-19 exposures as they emerge:

- This enables managers to quickly isolate exposed employees and implement policies.
- For example, they can identify a new outbreak at a site and then determine department(s).

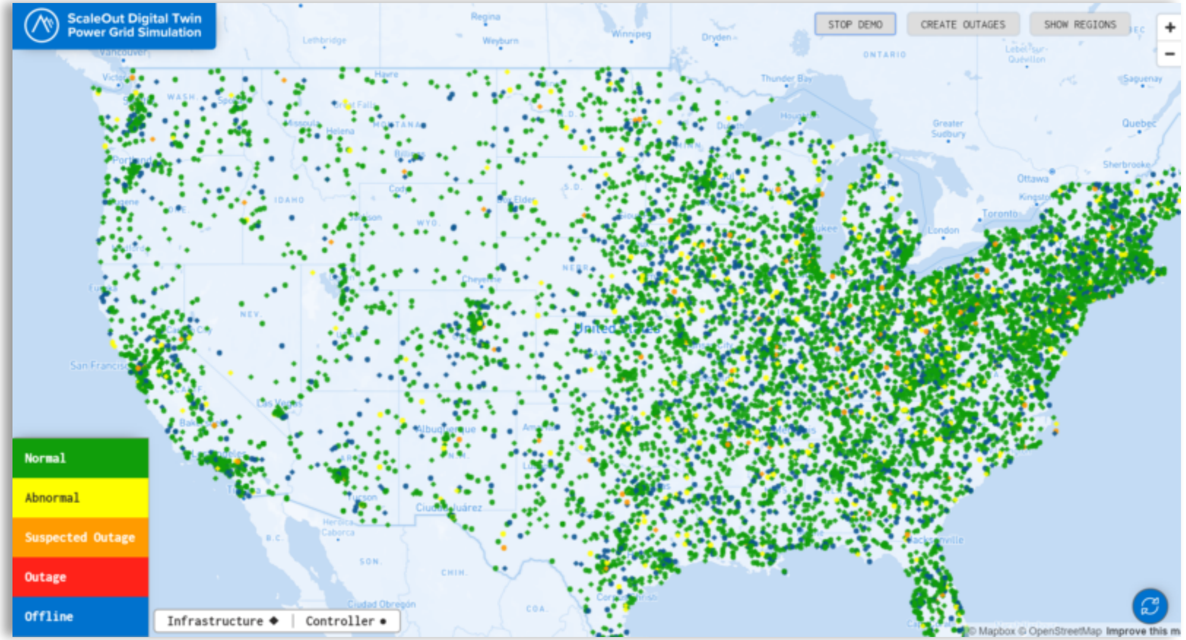




Example: Security Tracking for a Power Grid

Real-time digital twins can track nodes is a large power grid and detect intrusion points or emerging problems:

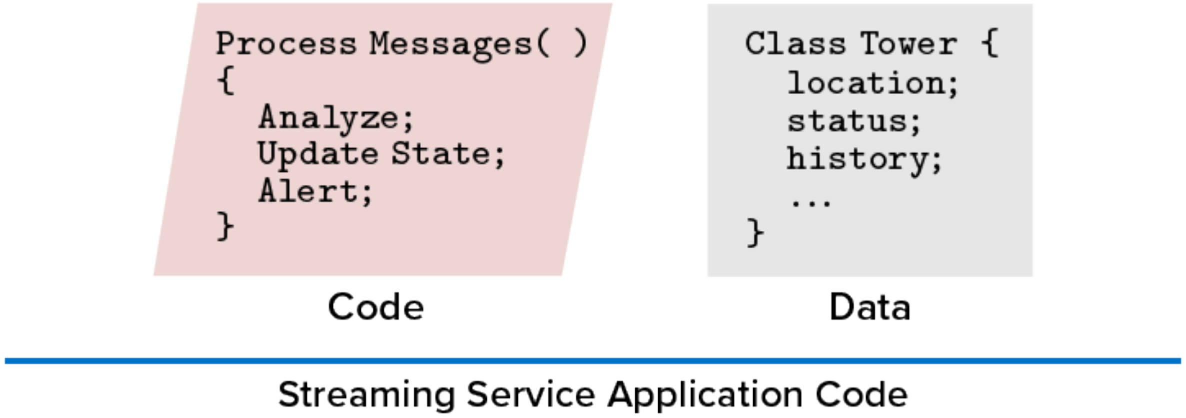
- Can introspect on intrusion events to predict likelihood of an attack.
- Can detect issues (e.g., overheating transformer) to predict likelihood of fire.
- Can create derived state describing the results of introspection (e.g., alert level).
- **Aggregate analytics** can give managers data needed for a strategic response.



Implementing a Digital Twin Model

Application developer creates a class to represent the state object and implements the ProcessMessages method:

- The platform correlates messages by source and runs the ProcessMessages method.
- The method accesses context from the state object and updates the object as needed.
- The method sends replies to the data source and send alerts as necessary.
- **The streaming platform can access the state object for aggregate analytics.**



Sample Code: State Object Definition

```
public class StatusTracker extends DigitalTwinBase {
    // State variables
    public String node_type;
    public String node_condition;
    public String region;
    public double longitude;
    public double latitude;
    // Derived state variables
    public int alert_level;
    public int minorIncidentCount;
    public int moderateIncidentCount;
    public int falseIncidentCount;
    public int severeIncidentCount;
    public int totalIncidents;
    public int totalResolvedIncidents;
    public boolean experiencingIncident;
    // Dynamic incident report list
    public List<IncidentReport> incidentList;}
```

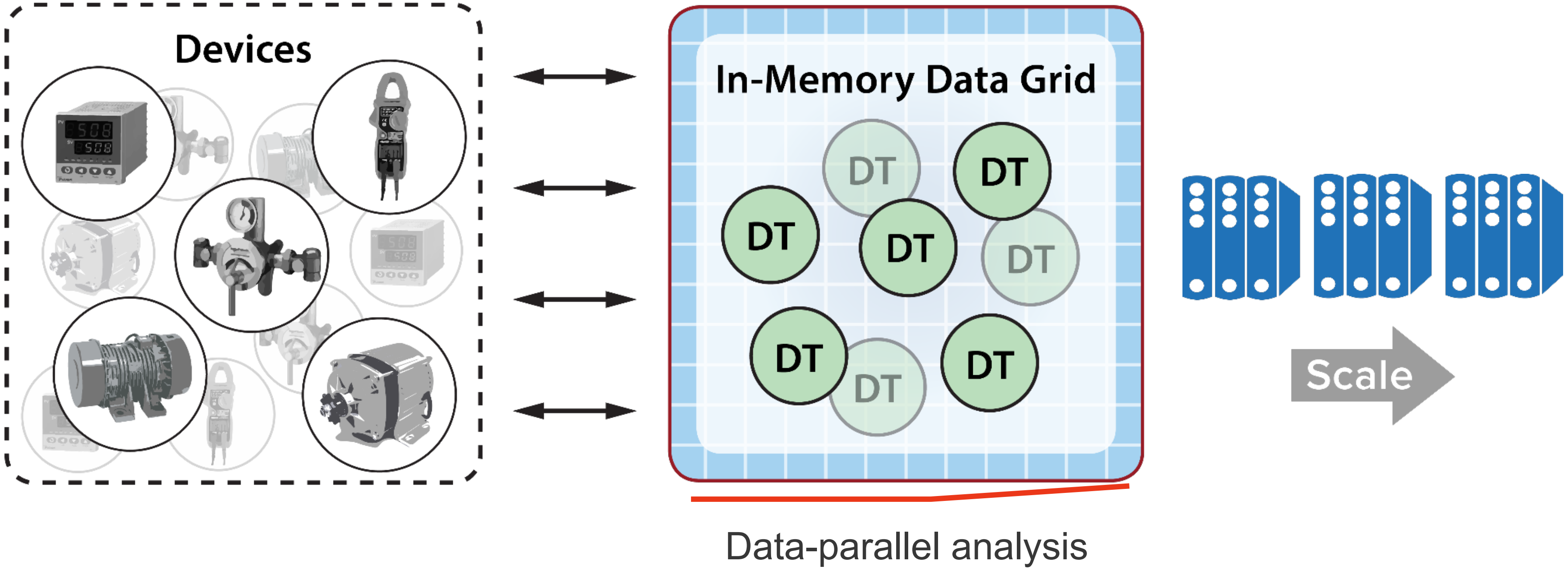
Sample Code: ProcessMessages Method

```
public ProcessingResult processMessages(ProcessingContext processingContext, StatusTracker
digitalTwin,
                                        Iterable<StatusTrackerMessage> messages) throws Exception
    // Iterate through the incoming messages:
    for(StatusTrackerMessage msg : messages) {
        // if the message indicates a moderate incident and this tracker has never had a severe
        // incident while the heuristic false incident ratio is greater than 50%, boost the alert
level:
        if(msg.moderateIncident() &&
            digitalTwin.getSevereIncidentCount() == 0 &&
            digitalTwin.getModerateIncidentCount() > 0 &&
            ((double)(digitalTwin.getFalseIncidentCount()/
                      digitalTwin.getModerateIncidentCount()) >= 0.5)) {
                digitalTwin.setAlertLevel(Constants.MODERATE+3);
                digitalTwin.incrementModerateEventCount();
                digitalTwin.setStatusTrackerCondition(msg.getNodeCondition());
            }
        // ... [additional rules]
    }
    return ProcessingResult.UpdateDigitalTwin;}
```

Running Real-Time Digital Twins on an IMDG

Real-time digital twins can be mapped to in-memory objects stored in an in-memory data grid (IMDG):

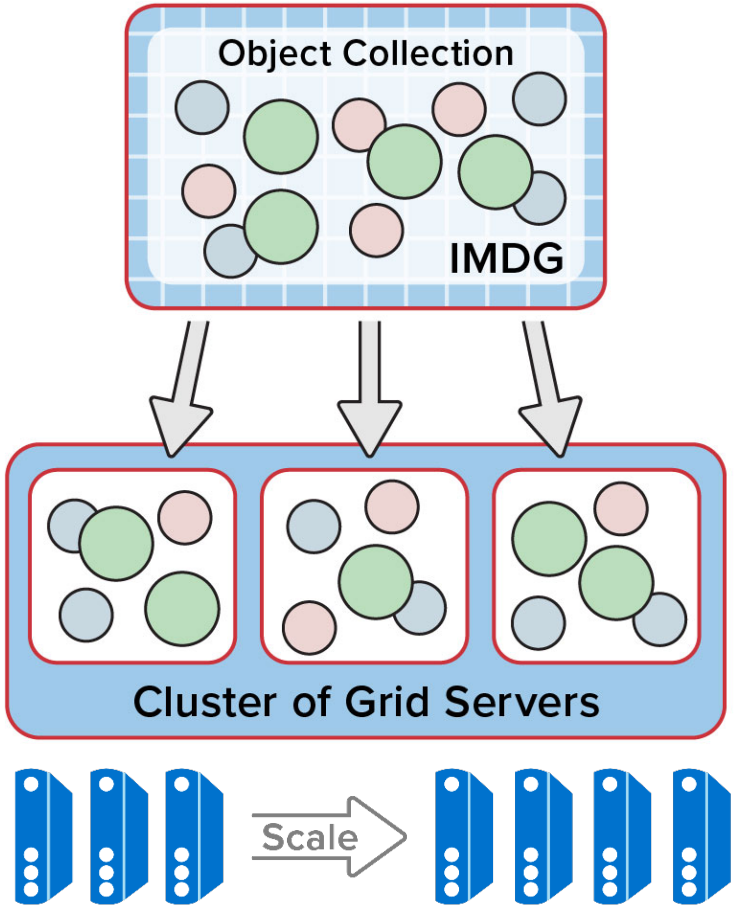
- The IMDG delivers messages to the servers on which the instances are hosted.



What Is an In-Memory Data Grid?

In-Memory Data Grid (IMDG) provides a scalable software platform for in-memory data storage and data-parallel computing.

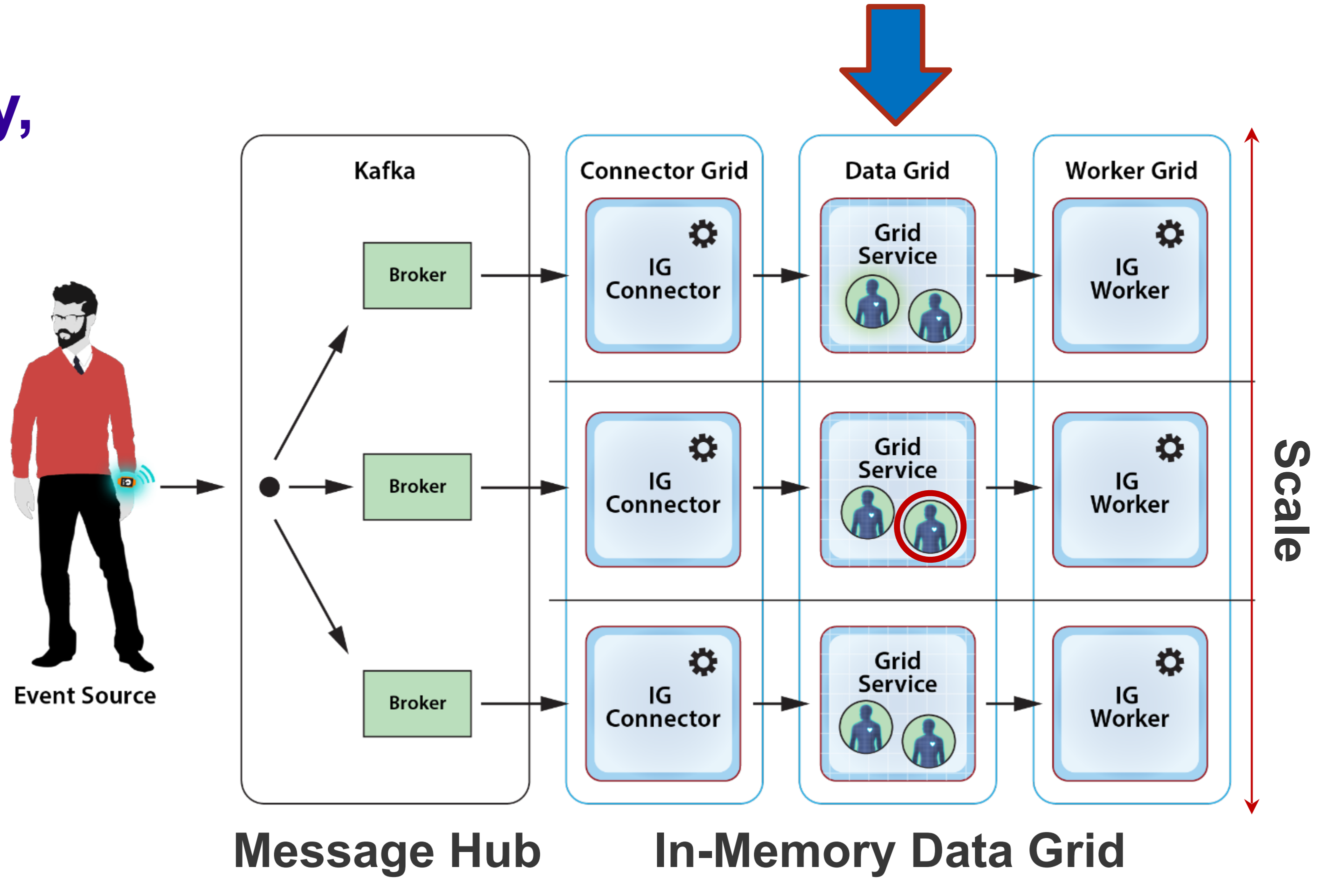
- Typically stores object-oriented data as key-value pairs distributed across a cluster of servers.
- Gives applications a global view of stored objects.
- Provides transparent scaling & high availability.
- Can include APIs for data-parallel computing.
- Takes advantage of multiple servers to provide high throughput and fast access times.



How an IMDG Runs a Digital Twin Model

IMDG uses a scalable software architecture for message delivery, object storage and message processing:

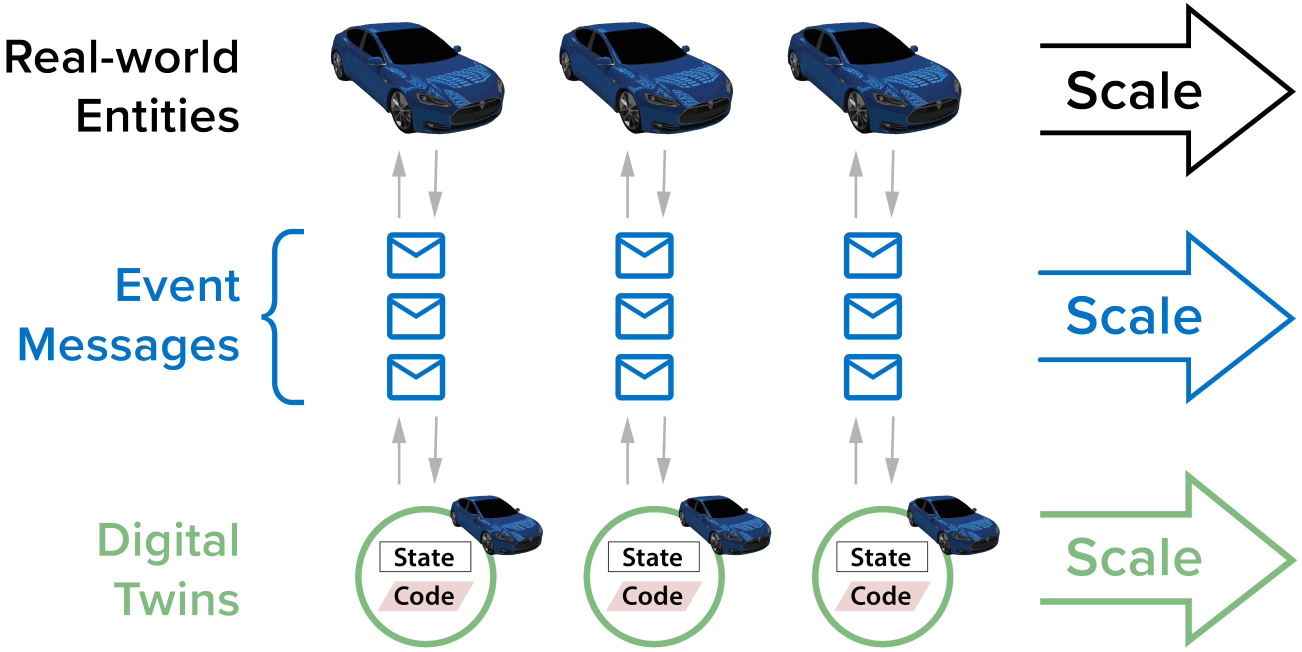
- Each server hosts three processes.
- Data grid's process hosts objects.
- Worker grid's process runs message-processing code.
- Connector grid's process delivers messages.



Advantages of Using an IMDG

IMDG enables fast message handling, scalable throughput, and high availability:

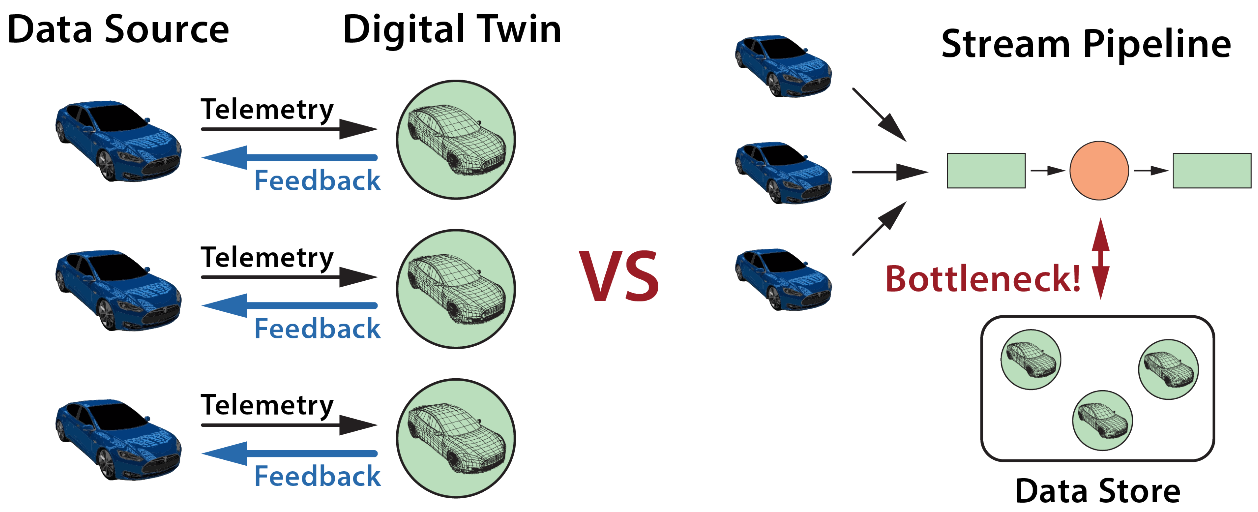
- Messages are processed where the instance objects are hosted to reduce data motion.
- The IMDG ensures reliable message delivery from the message hub.
- The IMDG transparently scales by adding servers to host more instances.



IMDG Reduces Network Bottlenecks

IMDG runs message-processing code on the host where the instance object is located:

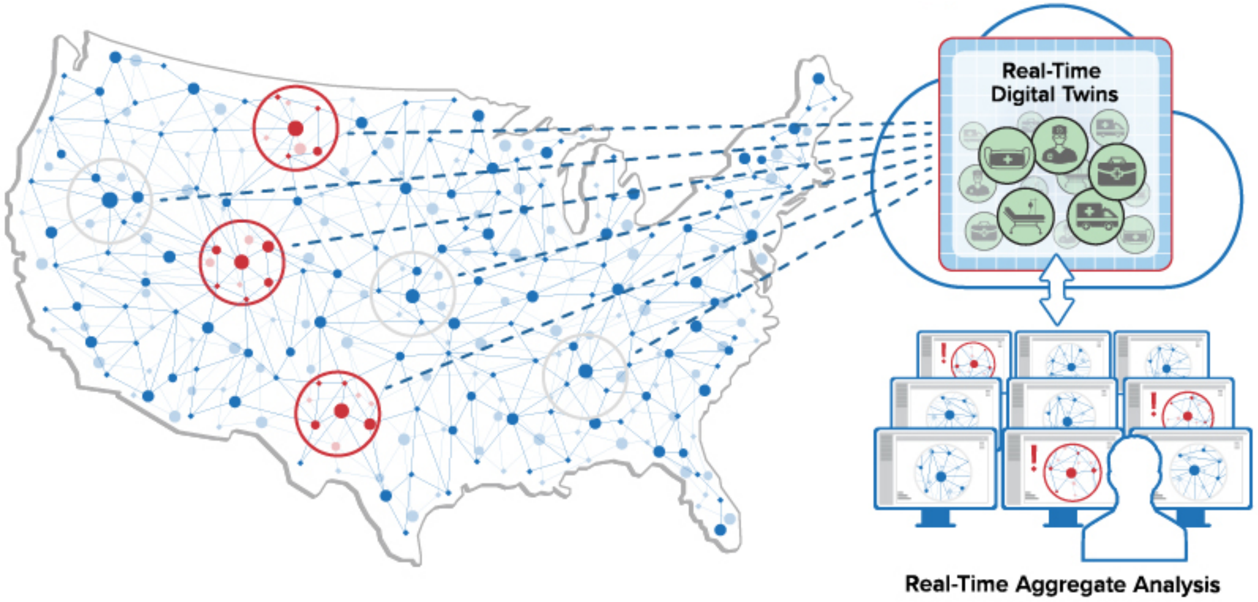
- This eliminates data motion to retrieve the state object.
- This reduces network bottlenecks and helps to maximize throughput scaling.
- A network transfer is still required to replicate updates.



Benefits of Real-Time Aggregate Analytics

Real-time aggregate analytics boost situational awareness:

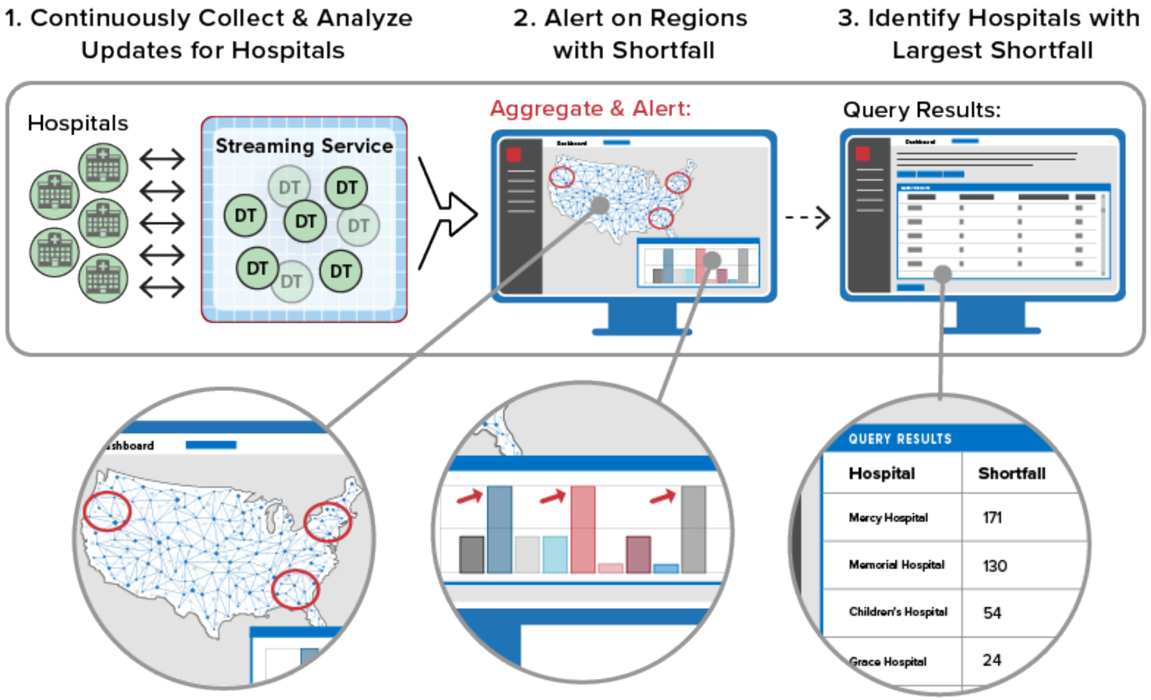
- Quickly pinpoint data sources which need attention.
- Allow managers to identify patterns and create strategies.
- For example, they can detect:
  - Multiple points of intrusion
  - Unusual shortfall of supplies in a region
  - Reaction to an ecommerce flash sale



Example of Aggregate Analytics

Aggregate analytics can help identify and react to emerging logistics issues. For example:

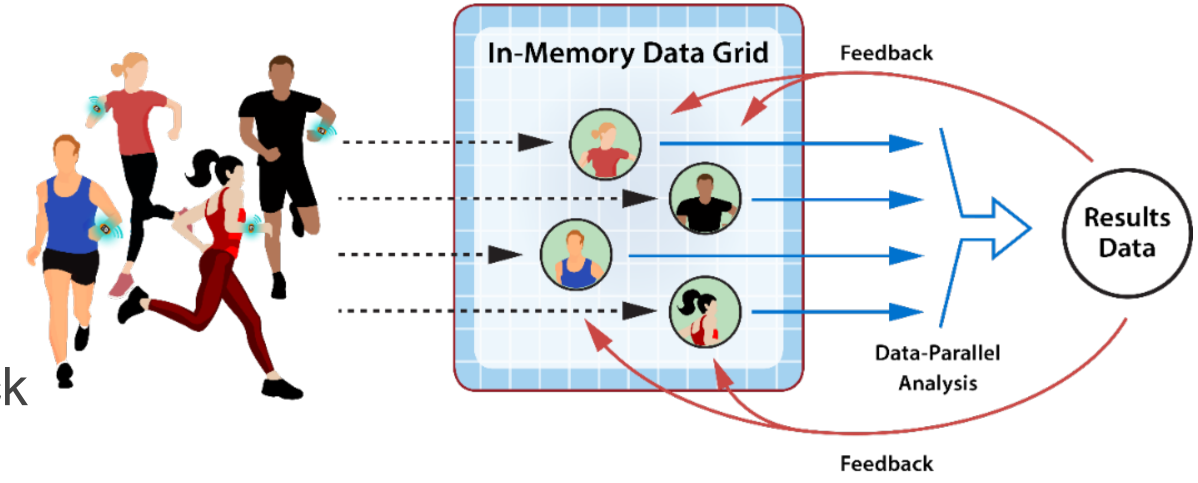
- Hospitals send periodic messages describing shortfall in supplies to their real-time digital twins.
- Real-time digital twins maintain curated shortfall data.
- Aggregate analytics immediately signals regions with highest needs.
- Query identifies hospitals with highest priority needs.



Implementing Aggregate Analytics

Real-time digital twins create a domain for aggregate analytics:

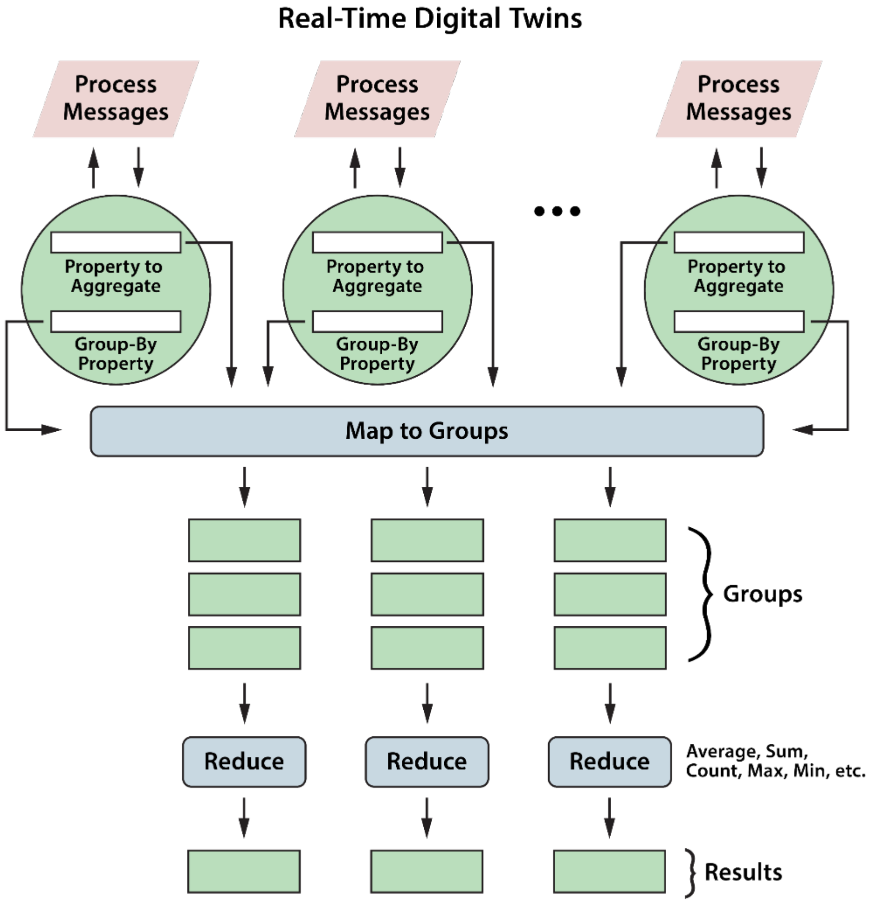
- Streaming pipelines and ad hoc systems must explicitly deliver data to a data lake for aggregate analysis.
- **Digital twin state objects provide an automatic domain for aggregate analysis.**
- Example: health device tracking
  - State objects collect statistics for each user.
  - Data-parallel analysis generates aggregate statistics.
  - These results can provide feedback to influence user behavior.



Integrating Aggregate Analytics with RTDTs

IMDG integrates MapReduce with digital twin state objects:

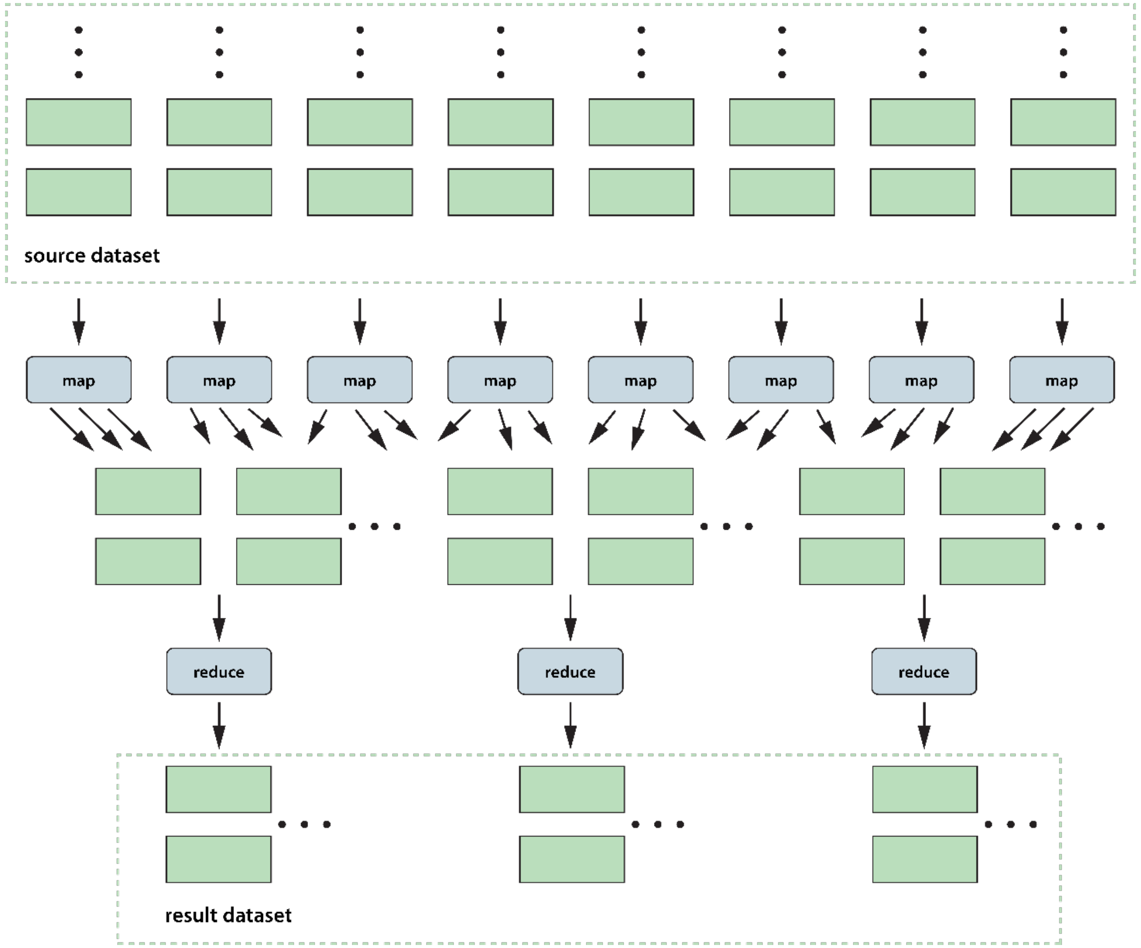
- Source data set is a selected property held in each digital twin instance's state object.
- Groups are defined by another selected state property (e.g., location, device type).
- The MapReduce operation extracts these properties concurrently with message processing.
- Aggregated results can be produced and refreshed every few seconds.



MapReduce for Aggregate Analytics

MapReduce runs in parallel to aggregate data in groups:

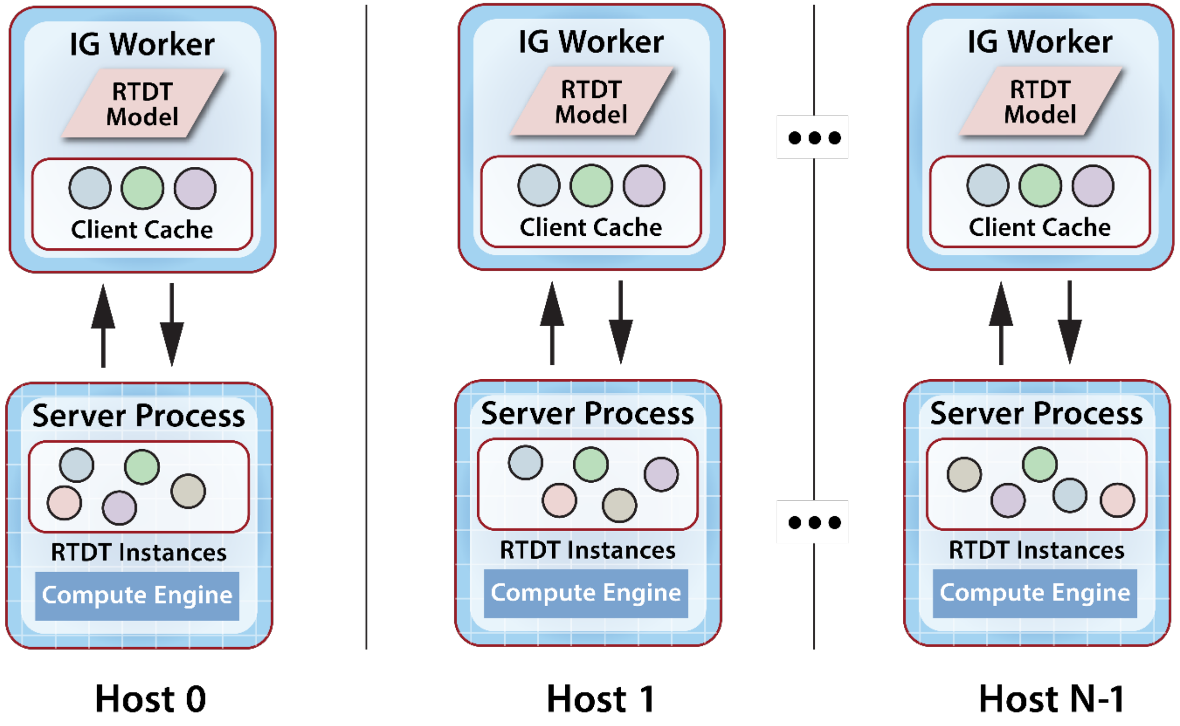
- Mappers partition input data into groups.
- Reducers combine data for each group using an aggregation operator.
- Mappers and reducers can run in parallel on each server using multiple threads.
- Results can then be merged and combined across multiple hosts.



Running MapReduce in an IMDG

MapReduce can be computed in a separate worker process on each IMDG server:

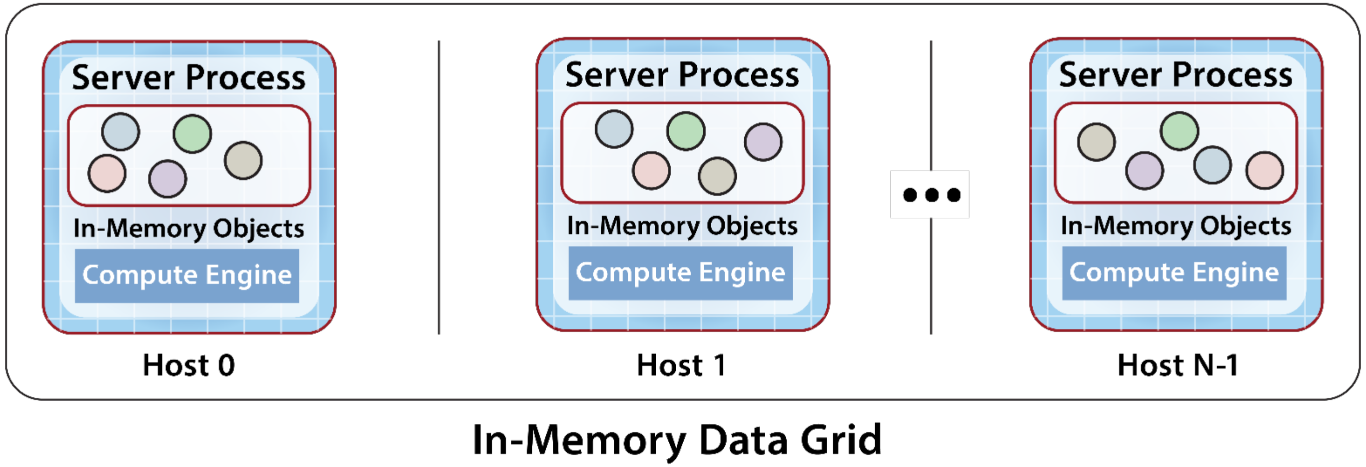
- Worker process could be a JVM or .NET runtime where the digital twin model runs.
- This provides flexibility in the MapReduce implementation.
- However, it requires objects to be transferred across an IPC connection for deserialization and analysis.



Running MapReduce in an IMDG (2)

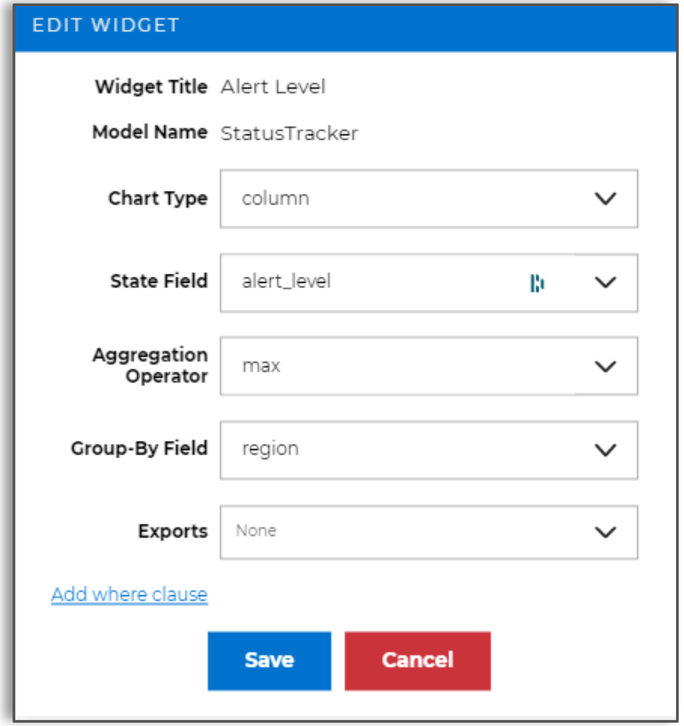
MapReduce can be computed within the IMDG service processes:

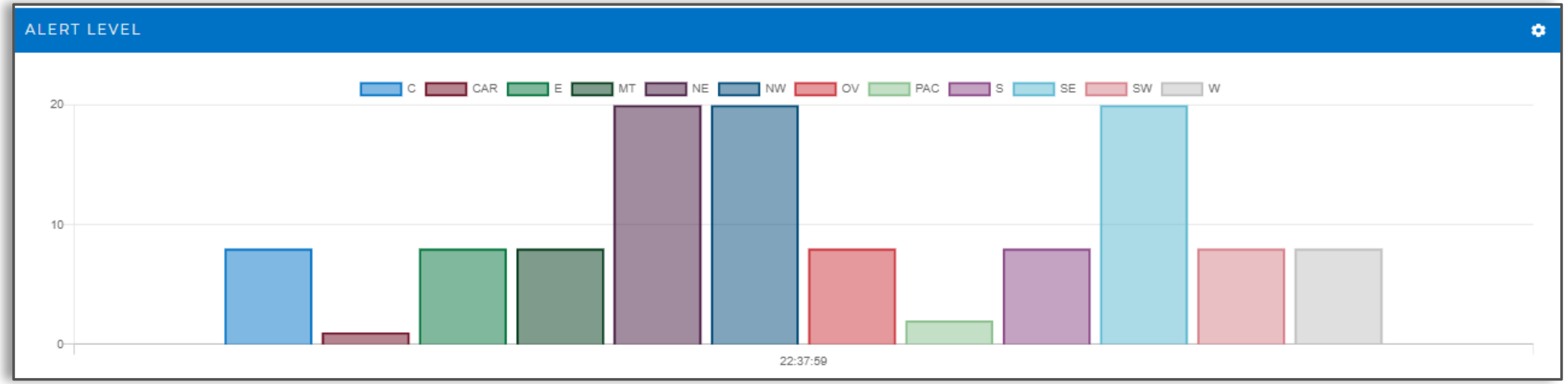
- This requires that digital twin instance objects use a known serialization format (e.g., JSON).
- It eliminates data motion except for merging results across servers.
- This provides the best throughput and reduces completion time.



Demo: Creating a MapReduce Widget

Demo shows how a visual widget can be created to perform continuous aggregate analytics on state information held in digital twin instances.





Widget Displaying Aggregate Analytics

Takeaways

- Real time digital twins enable streaming analytics to track large numbers of data sources.
- They offer several advantages over pipelined techniques, including context-based introspection and simplified design.
- They also provide a domain for aggregate analytics using properties in their state objects.
- IMDGs can host real-time digital twins, transparently scale message processing, and implement real-time aggregate analytics.
- These capabilities help managers maximize situational awareness.