

Online Visualization of Geospatial Stream Data using the WorldWide Telescope

Mohamed Ali[#], Badrish Chandramouli^{*}, Jonathan Fay^{*}, Curtis Wong^{*},
Steven Drucker^{*}, Balan Sethu Raman[#]

[#]Microsoft SQL Server, One Microsoft Way, Redmond WA 98052
{mali, sethur}@microsoft.com

^{*}Microsoft Research, One Microsoft Way, Redmond WA 98052
{badrishc, jfay, curtis.wong, sdrucker}@microsoft.com

ABSTRACT

This demo presents the ongoing effort to meld the stream query processing capabilities of Microsoft StreamInsight with the visualization capabilities of the WorldWide Telescope. This effort provides visualization opportunities to manage, analyze, and process real-time information that is of spatio-temporal nature. The demo scenario is based on detecting, tracking and predicting interesting patterns over historical logs and real-time feeds of earthquake data.

1. INTRODUCTION

Real-time stream data acquisition has been widely used in numerous applications such as network monitoring, telecommunications data management, security, manufacturing, and sensor networks. Besides the temporal nature of stream data, various sources provide real-time data with spatial characteristics – such as geographical locations and spatial extents such as point coordinates, lines, or polygons. Beyond stationary sensors, data is also continuously generated by moving objects thanks to advances in GPS and wireless technologies. The term “geo-streaming” refers to the ongoing effort in academia and industry to process and analyze stream data with geographic and spatial information.

Microsoft StreamInsight (StreamInsight, for brevity) [1, 2, 3, 4, 5] is a platform for stream query processing. StreamInsight monitors stream data to extract meaningful patterns and trends. StreamInsight analyzes and correlates data from multiple sources incrementally to yield low latency response times. The key features of StreamInsight are: a declarative query language with a temporal algebra that is agnostic of whether data is real-time or historical, the ability to natively handle stream disorder and provide early results, the ability to issue correctness guarantees on the output and the run-time query composability. Moreover, StreamInsight incorporates a powerful extensibility framework [13] that seamlessly integrates user defined modules into the

execution query pipeline. StreamInsight has been used in the past to process geospatial data, e.g., in traffic monitoring [4, 5].

On the other hand, the WorldWide Telescope (WWT) [9] enables a computer to function as a virtual telescope, bringing together imagery from the various ground and space-based telescopes in the world. WWT enables a wide range of user experiences, from narrated guided tours from astronomers and educators featuring interesting places in the sky, to the ability to explore the features of planets. One interesting feature of WWT is its use as a 3-D model of earth, enabling its use as an earth viewer.

This demo presents the ongoing work at Microsoft Research and Microsoft SQL Server groups to meld the value of stream query processing of StreamInsight with the unique visualization capabilities of WWT. The key contributions of this demo are:

- We integrate a streaming engine (StreamInsight) with a spatial visualization tool (WWT) in order to demonstrate new forms of visualization for spatio-temporal data.
- We allow the seamless incorporation of real-time and historical data querying/visualization using a single engine.
- We demonstrate the power of novel streaming queries for a specific spatio-temporal application – earthquake early detection and monitoring. We highlight the benefits of native disorder support in order to minimize event detection latency, while allowing eventual guarantees on result quality.

We note that the combination of real-time data processing with WWT provides new visualization opportunities to manage, analyze, and process spatio-temporal streaming information.

2. THE WORLDWIDE TELESCOPE

The WorldWide Telescope (WWT) [9] is powerful application that allows end users to explore the earth, the planets, the solar system and the sky. It serves as a tool and valuable resource for both astronomical and educational communities, as well as a way to make science interesting to the lay person. The WWT application brings together imagery from the best ground- and space-based observatories across the world. WWT uses a high-performance Microsoft Visual Experience Engine to allow seamless and interactive panning and zooming over terabytes of stitched-together imagery data. Images are loaded on-demand in order to conserve network bandwidth, with several optimizations to enhance the viewing experience. While the sky mode (for

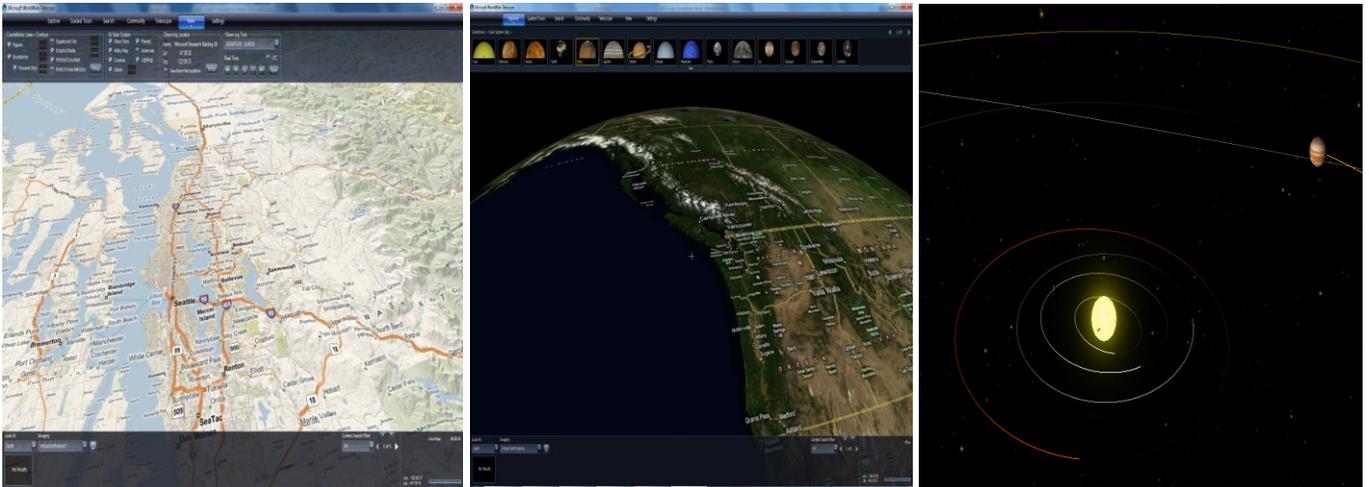


Figure 1. The WorldWide Telescope (WWT) as a viewer of the earth, the sky and the planets.

visualizing stars and galaxies) is the most popular, the underlying WWT engine is capable of supporting a wide range of image sources, including panoramas, planets, as well as user-provided equi-rectangular images with associated optional metadata.

A particularly interesting use of WWT is as a viewer of a 3-D model of earth, similar to Microsoft Virtual Earth and Google Earth. This application uses Virtual Earth’s satellite and map images, but is instead powered by the WWT engine. The aerial, street, and night views allow a seamless experience while navigating the globe. Figure 1 shows a screenshot of WWT in action as an earth and a sky viewer.

The World Wide Telescope Software Development Kit (SDK) enables the customization of data and the development of customized interfaces through an API, such as allowing us to import our own data into the software. Of particular interest to our demonstration proposal is the ability to feed temporal (i.e., time-series) data such as logs of earthquake data including latitude, longitude, label, magnitude, and depth below the surface. In this case, the WWT visualizer allows us to “travel through time” while visualizing earthquakes along with their magnitudes.

3. MICROSOFT STREAMINSIGHT

A streaming system allows applications to execute long-running continuous queries (CQs) that monitor and process data streams in real time. Our demo focuses on Microsoft StreamInsight [3, 4, 5] which is based on the CEDR [1, 2] research project. Some distinguishing features of StreamInsight include a deterministic underlying temporal algebra, a powerful LINQ-based querying interface [7, 8], debuggability, native support for out-of-order events, high availability, scalability, speculative event processing, and a powerful extensibility framework. Out of the mentioned features, we select few features that are relevant to our demo to discuss next: StreamInsight’s temporal algebra and extensibility model; other issues such as time progress and state cleanup are covered in [2, 6].

3.1 Disorder Handling and Compensations

Each event consists of an event generation time, and a duration, which indicates the period of time over which an event can

influence output. We capture these by defining the control parameter associated with events as $\langle LE, RE \rangle$, where the interval $[LE, RE)$ specifies the period (or lifetime) over which the event contributes to output. The left endpoint (LE) of this interval, also called start time or event timestamp, is the application time of event generation. Assuming the event lasts for x time units, the right endpoint of an event (called end time), is $RE = LE + x$.

Compensations. StreamInsight can natively process disordered events, thus avoiding the need to buffer data at the source. Disorder is very common, e.g., due to network delays, merging data from multiple sources, etc. Native disorder processing is particularly useful in spatio-temporal applications such as earthquake detection, where an early detection is often very useful, even at the expense of potential false-positives. In order to compensate for results that are deemed incorrect at a later point, StreamInsight supports compensations (or corrections) for earlier reported events, using the notion of revisions [11]. A revision indicates a modification (or deletion) of the lifetime of an earlier event. This is supported by an optional third parameter $REnew$, that indicates the new right endpoint of the corresponding event.

Canonical History Table (CHT). This is the logical representation of a physical stream [1, 2]. Each entry in a CHT consists of a lifetime and the payload. Thus, StreamInsight models a data stream as a time-varying relation, and all operators have deterministic well-understood semantics based on their effect on the CHT. Time is a part of the application-provided event, and thus query results are the same regardless of whether we are processing real-time or historical data, which allows deterministic back-testing [12].

3.2 Extensibility Framework

The extensibility framework in StreamInsight [13] connects the user defined module (UDM) writer (who writes the extension libraries) and the query writer (who writes StreamInsight queries that may invoke UDMs). The extensibility framework provides flexibility and efficiency to both the UDM writer and the query writer. It allows domain experts to seamlessly integrate years of expertise in a specific domain with the processing capabilities of a streaming system. A typical use case of the extensibility

framework is the development of user defined pattern detection algorithms [10]. Of particular interest to this demo is a set of pattern detection UDMs for earthquake data.

4. System Overview

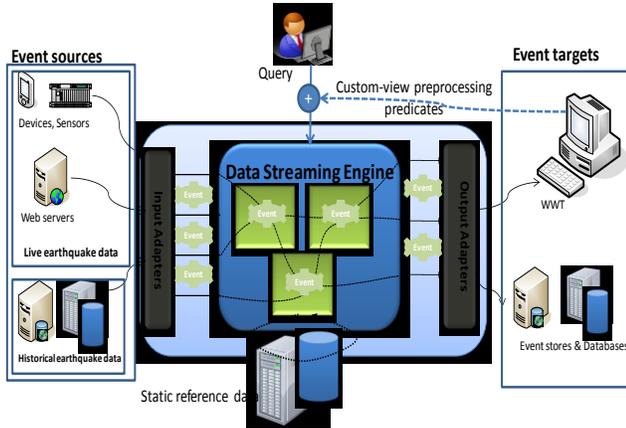


Figure 2. System overview.

Figure 2 provides an overview of the proposed system. The data streaming engine constitutes the heart of the system. The engine imports historical data from database systems and accepts real time feeds from sensors, device and web servers. All input data are fed to the engine through the appropriate input adapters. Input

adapters have the ability to interact with the stream data source and to push the stream events to the streaming engine. The engine processes all queries issued by the user and streams the resultant output to the consumer through output adapters. In this demo, the output is sent to the WWT and “probably” stored in databases for archiving purposes. Also, the streaming engine has the ability to access and join reference data, e.g., maps, road information and topological data, with stream data.

Focusing on the communication between the streaming engine and the WWT, we notice a bidirectional relationship:

First, the streaming engine (a) streams output events and (b) drives the observing time of the WWT according to the application time. Thanks to the disorder handling and compensations (as described in Section 3.1), WWT users can navigate in the future (through the observing time pane – Figure 3) to browse tentative but early results. As time proceeds and as more input tuples are received, the results are stabilized.

Second, the WWT has the ability to stitch a custom-view preprocessing query in front of the user issued query to: (a) filter out data points that are outside the visualization range, and (b) to adjust the sampling rate of the output according to the zoom level.

The purpose of the custom-view preprocessing query is to inject filters and sampling operators in the query pipeline. The engine pushes these filters down the query plan to the input adapters to filter output tuples that consume processing resources and that produce output that is outside the user’s visualization area. Similarly, as the user zooms out, the sampling rate is reduced to deliver good-enough images. As the user zooms in to a specific

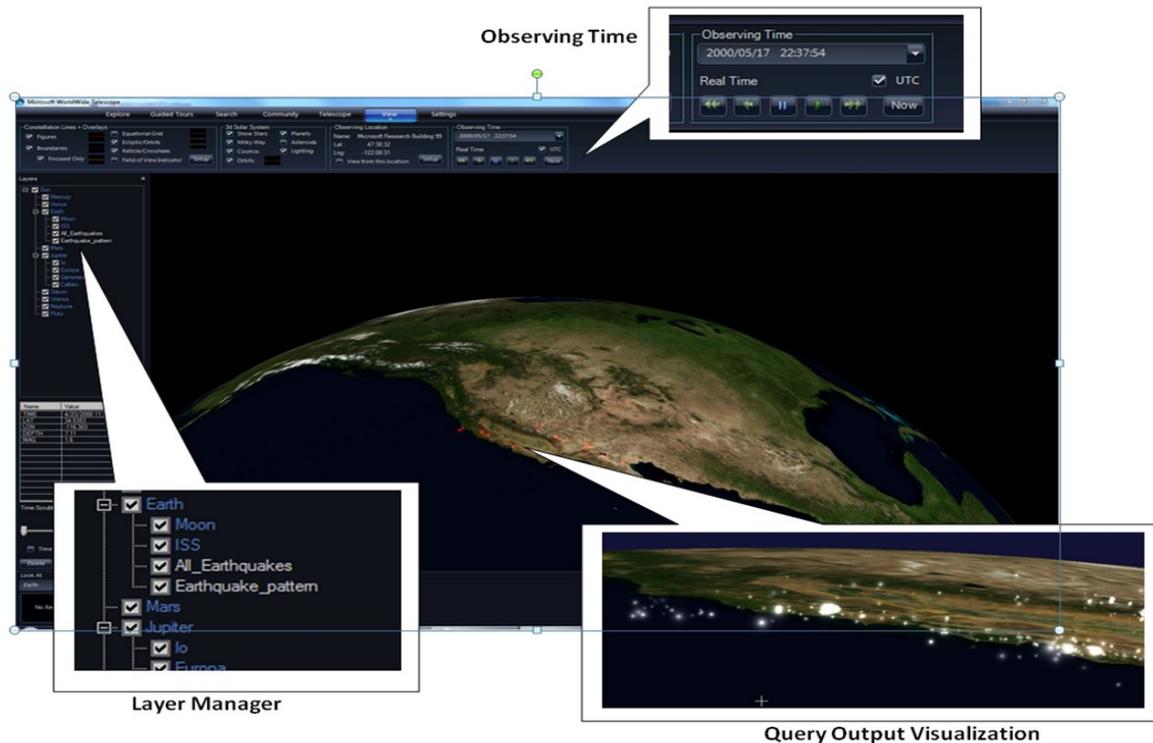


Figure 3. A screenshot of the WWT visualizing stream data.

region, the sampling rate is increased to deliver more detailed output in that region.

5. DEMO SCENARIO

The demo scenario has been prepared to deliver five key messages. We summarize these messages as follows:

- StreamInsight processes, analyzes and mines streams of “real time” data.
- StreamInsight processes, analyzes, and mines for interesting patterns in “historical” temporal data.
- StreamInsight is a full-fledged relational query processor, where queries that contain operators like select, project, join, etc are executed efficiently.
- StreamInsight is an extensible system where user-specific logic and domain expertise are integrated seamlessly in the query pipeline.
- The WWT visualization capabilities, the real time processing capabilities and the knowledge gained from historical data analysis, help take better decision about the future.

To achieve the above goals, the demo presents several StreamInsight queries that are issued against historical logs of earthquake data that have been accumulated over years. These queries range from simple queries that filter earthquake data based on location and magnitude to complex queries that involve pattern detection of interesting earthquake patterns. The queries include both relational operators as well as user defined operators built by domain experts. Moreover, the demo features a set of queries that run against real time earthquake data to detect interesting patterns, where the notion of “interesting” is defined based on the past experience with earthquake patterns that looked indicative to major disasters in the past.

Figure 3 represents the graphical interface of the WWT. We highlight that the interface has (1) a “layer manager” pane, where the output of each query is visualized in a separate layer. The output of several queries can be layered on top of each other to contrast and differentiate the results, (2) an “observing time” pane, which is an essential part in real time as well as historical temporal data. With its forward and backward buttons, the user can travel in the past as well as in the future (thanks to the speculative nature of StreamInsight), and (3) a “query output

visualization” pane, where the magnitude and the depth of the earthquake (with respect to the earth surface) are visualized using color schemes at the earthquake’s observed latitude and longitude.

6. REFERENCES

- [1] Roger S. Barga, Jonathan Goldstein, Mohamed Ali, and Mingsheng Hong. Consistent Streaming Through Time: A Vision for Event Stream Processing. In Proceedings of CIDR, 412-422, 2007.
- [2] Jonathan Goldstein, Mingsheng Hong, Mohamed Ali, and Roger Barga. Consistency Sensitive Streaming Operators in CEDR. Technical Report, MSR-TR-2007-158, Microsoft Research, Dec 2007.
- [3] Mohamed Ali et al. Microsoft CEP Server and Online Behavioral Targeting. VLDB 2009.
- [4] Mohamed Ali, Badrish Chandramouli, Balan Sethu Raman, Ed Katibah. Spatio-Temporal Stream Processing in Microsoft StreamInsight. IEEE Data Eng. Bull. 33(2): 69-74 (2010).
- [5] Seyed J. Kazemitabar, Ugur Demiryurek, Mohamed Ali, Afsin Akdogan, Cyrus Shahabi, Geospatial Stream Query Processing using Microsoft SQL Server StreamInsight, In VLDB, 2010.
- [6] B. Chandramouli, J. Goldstein, and D. Maier. On-the-fly Progress Detection in Iterative Stream Queries. In VLDB, 2009.
- [7] Paolo Pialorsi, Marco Russo. Programming Microsoft LINQ, Microsoft Press, May 2008.
- [8] Microsoft LINQ. <http://tinyurl.com/42egdn>.
- [9] World Wide Telescope project. <http://www.worldwidetelescope.org/>.
- [10] B. Chandramouli, J. Goldstein, and D. Maier. High-Performance Dynamic Pattern Matching over Disordered Streams. In VLDB, 2010.
- [11] E. Ryvkina et al. Revision processing in a stream processing engine: A high-level design. In ICDE, 2006.
- [12] Alex Raizman et al. An Extensible Test Framework for the SQL Server StreamInsight Query Processor, In DBTest, 2010.
- [13] Mohamed Ali, Badrish Chandramouli, Jonathan Goldstein, Roman Schindlauer. The Extensibility Framework in Microsoft StreamInsight, In ICDE, 2011.