15 billion connected devices by 2015
The data deluge

AND HOW TO HANDLE IT: A 14-PAGE SPECIAL REPORT
concepts
Event processing is a method of tracking and analysing (processing) streams of information (data) about things that happen (events)

http://en.wikipedia.org/wiki/Complex_event_processing
Complex event processing combines data from multiple sources to infer events or patterns that suggest more complicated circumstances.

http://en.wikipedia.org/wiki/Complex_event_processing
men in suits
woman in white dress
church bells ringing
throwing of rice
It’s a wedding!
Isn’t this just a database application?
how many sail powered yachts in marina?

1. Walk out to marina
2. Count objects that are yachts and have sails

select count(*) from Marina where type="yacht" and powered="sail"
how many red cars have passed in last hour?

1. Pull all cars off highway into holding lot.
2. Leave each car there for at least 60 min. Keep track of who can leave.
3. Update count of vehicles everytime one enters if it is red and a car.

job - loading records in HoldingLot
loop - select count(*) from HoldingLot where type="car" and colour="red"
job - evicting records from HoldingLot
Different kinds of questions require different techniques for answering them.
<table>
<thead>
<tr>
<th></th>
<th><strong>database</strong></th>
<th><strong>complex event processing</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>queries</td>
<td>ad hoc on static data</td>
<td>continuous standing queries</td>
</tr>
<tr>
<td>latency</td>
<td>seconds</td>
<td>milliseconds</td>
</tr>
<tr>
<td>data rate</td>
<td>hundreds per second</td>
<td>Tens of thousands per second</td>
</tr>
</tbody>
</table>
Complex Event Processing

- Processing and querying of event data streams
- Data queried while “in flight”
- May involve multiple concurrent event sources
- Works with high data rates
- Aims for near-zero latency
Complex Event Processing

Identify from seemingly unrelated events:

- Patterns
- Relationships
- Gaps (expected events that did not occur)
- Abstractions
Verticals

- Financial services:
  - Risk analytics
  - Algorithmic trading

- Industrial automation:
  - Process control and monitoring
  - Historian based process analysis

- Security:
  - Anomaly/fraud detection

- Web analytics:
  - Behavioural targeting
  - Clickstream analytics
  - Customer relationship management

- Business intelligence:
  - Reporting
  - Predictive analytics
  - Data cleansing
streaminsight
a powerful platform from Microsoft that you can use to develop and deploy complex event processing (CEP) applications that derive immediate business value from high-throughput streams of data.

Visual Studio 2010/2012, .NET 4, C#, LINQ, Rx
Gartner’s 3Vs of Big Data – Volume, *Velocity* and Variety
Event is basic unit of data processed by StreamInsight
Event

- Structure
  - Header, Payload

- Kind
  - INSERT, CTI

- Model
  - Interval, Point, Edge
Event - Structure

- Header
  - Metadata (event kind, timestamps)
  - Timezone aware

- Payload
  - .NET data structure
  - Must be present
  - Event page size – 16k

```csharp
public class Toll
{
    public string TollId { get; set; }
    public double TollAmount { get; set; }
    public long VehicleCount { get; set; }
}
```

Event - Kind

- INSERT
  - Start Time, End Time, Payload

- CTI (common time increment)
  - Start Time
  - Special punctuation event – completeness and low latency
  - Without CTI, no output generated by queries
Event - Model (Interval)

- Payload valid for period of time
- Start Time (inclusive), End Time (exclusive)

<table>
<thead>
<tr>
<th>event kind</th>
<th>start</th>
<th>end</th>
<th>payload</th>
</tr>
</thead>
</table>
Event - Model (Interval)

Examples

• width of an electronic pulse
• duration of (validity of) an auction bid
• stock ticker activity in which the bid price for the stock is valid for a specific time period
Event - Model (Point)

- Payload valid for a single point in time
- Start Time
- Implicit End Time (+1 .NET tick, equivalent to 100 nanoseconds)

<table>
<thead>
<tr>
<th>event kind</th>
<th>start</th>
<th>end</th>
<th>payload</th>
</tr>
</thead>
</table>
Event - Model (Point)

Examples

• meter reading
• arrival of an email
• user Web click
• stock tick
• entry into the Windows Event Log
**Event - Model (Edge)**

- Payload valid for interval of time
- EdgeTime
- Edge Type (START | END)

<table>
<thead>
<tr>
<th>event kind</th>
<th>edge type</th>
<th>start</th>
<th>end</th>
<th>payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT</td>
<td>start</td>
<td>t0</td>
<td>DateTimeOffset.MaxValue</td>
<td>a</td>
</tr>
<tr>
<td>INSERT</td>
<td>end</td>
<td>t0</td>
<td>t1</td>
<td>a</td>
</tr>
<tr>
<td>INSERT</td>
<td>start</td>
<td>t1</td>
<td>DateTimeOffset.MaxValue</td>
<td>b</td>
</tr>
<tr>
<td>INSERT</td>
<td>end</td>
<td>t1</td>
<td>t2</td>
<td>b</td>
</tr>
</tbody>
</table>
Event - Model (Edge)

Examples

- Windows processes
- trace events from Event Tracing for Windows (ETW)
- a web user session
query
Temporal stream implemented via the IQStreamable<> interface
Query

Processed via StreamInsight LINQ

- Projection
- Filtering
- Joins (Inner, Cross, Anti-Semi-Join)
- Group and Apply
- Timestamp modification
windows
windows allow us to perform set-based computation (aggregation) over subsets of events that fall within some period of time
Tumbling Window

- Gapless, non-overlapping windows
- Defined by 2 timespans
  - Hop size $H$
  - Window size $S$
  - $H = S$
Every 3 minutes, count the number of vehicles processed at the toll station since the last result.

Report the result at a point in time, at the end of the 3 minute window.
Every 3 minutes, report the number of vehicles processed that were being processed at some point during that period at the toll station since the last result.

Report the result at a point in time, at the end of the 3 minute window.
demo
Hopping Window

- Tumbling window is specialised case
- Can create overlapping windows

- Defined by 2 timespans
  - Hop size H
  - Window size S
  - If $H < S$ overlapping windows
Report the count of vehicles being processed at some time over a 3 minute window, with the window moving in one minute hops.

Provide the counts as of the last reported result as of a point in time, reflecting the vehicles processed over the last 3 minutes.
demo
Snapshot Window

- Dynamic
- Event driven
- Defined by
  - Start and end times of events
Find the most recent toll generated from vehicles being processed at each station over a one minute window reporting the result every time a change occurs in the input.
demo
extensibility
User defined functions (UDF)

- Per event computations
- Used in filter (commonly)
- Used in project
```csharp
var query = from e in tollStream
            where 0 == e.Tag.Length || TagInfo.IsLostOrStolen(e.Tag) || TagInfo.IsExpired(e.Tag)
            select new TollViolation
            {
                LicensePlate = e.LicensePlate,
                Make = e.Make,
                Model = e.Model,
                State = e.State,
                Tag = e.Tag,
                TollId = e.TollId,
                Vin = GetVinFromLicensePlate(e.LicensePlate)
            };
```
User defined aggregate (UDA)

- Compute aggregate over set of events
- Could be expensive
- Sum, count, Avg are incremental, UDAs are not
var query = from win in tollStream.TumblingWindow(TimeSpan.FromMinutes(3))
    select win.UserDefinedAggregate<TollReading, OutOfStateVehicleRatio, float>(null);

public class OutOfStateVehicleRatio : CepAggregate<TollReading, float>
{
    public override float GenerateOutput(IEnumerable<TollReading> tollReadings)
    {
        float tempCount = 0; float totalCount = 0;
        foreach (var tollReading in tollReadings)
        {
            totalCount++;
            if (tollReading.State != "NY") { tempCount++; }  
        }
        return tempCount / totalCount;
    }
}
User defined operator (UDO)

- Define event transformers
- Operation not covered by
  - Join
  - Union
  - Group
  - Apply
  - Project
```csharp
var query = from win in tollStream.TumblingWindow(TimeSpan.FromHours(1))
    from e in win.UserDefinedOperator(() => new VehicleWeights())
    select e;

public class VehicleWeights : CepTimeSensitiveOperator<TollReading, VehicleWeightInfo>
{
    private double weightcharge = 0.5

    public override IEnumerable<IntervalEvent<VehicleWeightInfo>> GenerateOutput(
        IEnumerable<IntervalEvent<TollReading>> events, WindowDescriptor windowDescriptor)
    {
        List<IntervalEvent<VehicleWeightInfo>> output = new List<IntervalEvent<VehicleWeightInfo>>();

        // Identify any commercial vehicles in this window for the given window duration
        foreach (var e in events.Where(e => e.StartTime.Hour >= 0 && e.Payload.VehicleType == 2))
        {
            // create and populate output interval event
            IntervalEvent<VehicleWeightInfo> vehicleWeightEvent = CreateIntervalEvent();
            vehicleWeightEvent.StartTime = e.StartTime; vehicleWeightEvent.EndTime = e.EndTime;
            vehicleWeightEvent.Payload = new VehicleWeightInfo
            {
                LicensePlate = e.Payload.LicensePlate, Weight = e.Payload.VehicleWeight,
            }

            // here is the interesting part; note how the output is dependent on
            // the start and end timestamps of the input event. The weight charge
            // is a function of the rush hour definition, the weigh charge factor
            // and the vehicle tonnage itself

            output.Add(vehicleWeightEvent);
        }

        return output;
    }
}
project “austin”
Why cloud?

- Data is not local
  - Event data is already in cloud
  - Event data is globally distributed

- Benefits
  - Reduced TCO
  - Elastic scale-out
  - Service, not infrastructure
demo
Resources

- **MSDN - Microsoft StreamInsight 2.1**

- **CodePlex - StreamInsight Samples**

- **A Hitchhiker's Guide to StreamInsight Queries**
Blogs

• StreamInsight Team
  http://blogs.msdn.com/b/streaminsight/

• Mark Simms
  http://blogs.msdn.com/b/masimms/archive/tags/streaminsight/

• J Sawyer
  http://www.devbiker.net/category/StreamInsight.aspx

• Richard Seroter
  http://seroter.wordpress.com/category/streaminsight/