Databasebelasting kun je beheersen met SSSI

Continuous Event Processing met SQL Server SI

Paul van Wingerden

Met behulp van Microsoft StreamInsight is het mogelijk om 'Continuous Event Processing' applicaties te schrijven. Met de toenemende hoeveelheden data die door allerlei systemen en sensoren geproduceerd worden is het vaak moeilijk om deze data op een tijdige manier te verwerken. StreamInsight is ontworpen voor systemen die met een zeer lage latency en een hoge throughput events moeten kunnen verwerken.

Verwerken van miljoenen events per minuut vormt een enorm belasting voor het systeem. De disk reads en writes rijzen de pan uit als je dit niet optimaal organiseert. StreamInsight is een belangrijk hulpmiddel om de hoge throughput van events te stroomlijnen.

Voorbeelden van systemen met een zeer lage latency en een hoge throughput van events zijn:

- Sensor netwerken: Een smart power grid waarbij afnemers van stroom een intelligente stroommeter hebben die periodiek het verbruik afgeeft. Dit kan gekoppeld worden aan wind- en zonnegegevens zodat vraag een aanbod beter afgestemd kunnen worden;
- Financiële handels applicaties (algorithmic trading);
- Schepen: Een schip zit vol met honderden tot duizenden sensoren die onder andere de stabiliteit bewaken en corrigeren. Hiermee kan onder andere brandstof bespaard worden;
- Industriële processen zoals bijvoorbeeld een staalsmelter bij een hoogoven;
- Fabrieken waar lopende banden zijn waar bijvoorbeeld auto's geassembleerd worden;
- Webfarm waarvan de events gebruikt worden ten behoeve van de analyse van het gebruik van de site;

In deze systemen moet meestal worden gezocht naar patronen in de data. Vaak moeten events die uit vele verschillende bronnen (er kunnen meerdere producenten zijn voor de data) komen, eerst gecorreleerd worden. Daarna kan de data worden gemonitord.

Wat zijn de huidige problemen?

In het verleden heb ik aan een systeem meegewerkt (Microsoft Mediaroom) waarbij extreem grote aantallen events geproduceerd konden worden. Het is technologie waarmee digitale televisie gedistribueerd wordt. De schaalgrootte waarmee rekening gehouden moet worden zijn miljoenen settopboxen, die allemaal veel events produceren. Niet alle events zijn even belangrijk, het veranderen van kanaal is minder belangrijk dan het aanschaffen van een video. Om deze hoeveelheden events te kunnen verwerken moest er veel technologie gebouwd worden die nu zo out-of-the box beschikbaar is met StreamInsight. De output van dit systeem voedt onder andere een management dashboard met KPIs.

Nou kan je je afvragen waarom voor het analyseren van bijvoorbeeld web traffic een hele nieuwe tool gebruikt zou moeten worden. Je kunt immers met SQL, eventueel in combinatie met MDX (olap) ook zeer zware query's draaien? Een van de uitdagingen die je met bovengenoemde aanpak hebt is het 'realtime' aspect. De data moet eerst weggeschreven worden naar een tabel of een file, voordat die verwerkt kan worden. Een voorbeeld van verwerken kan zijn om een aggregatie zoals SUM of COUNT bij te houden. Deze waarden kunnen pas uitgerekend worden als alle data op disk staan. Een ander voorbeeld is als je in de stroom van events zoekt naar een bepaalde waarde, bijvoorbeeld een overschrijding die binnen een tijdsinterval een aantal maal moet voorkomen. StreamInsight stelt je in staat om in de stream hiernaar te zoeken.

Stel voor dat je miljoenen events per minuut moet kunnen verwerken en die events stop je eerst in een tabel (bijvoorbeeld via BCP) om vervolgens deze events te verwerken. De resulterende data gaat hierna naar een warehouse. Door gelijk de events te verwerken in plaats van ze eerst in een file weg te schrijven, die vervolgens te bulkloaden (BCP), dan te querien en de resultaten weer weg te schrijven heb je maar zo drie maal de hoeveelheid disk reads en writes te pakken. Een capaciteits berekening indertijd leerde dat alleen voor het pre-processen we een van de grootste servers die toen bestonden moesten aanschaffen alleen om de events te kunnen verwerken. Dit nog onafhankelijk van de overige hardware die voor dit soort systemen benodigd waren. Een van de voordelen van een in-memory of (in flight) methode zoals StreamInsight is dat je gigantische hoeveelheden IO bespaard. StreamInsight helpt je dus om deze hoeveelheden events beheersbaar te maken.
In dit artikel wordt en voorbeeld gegeven hoe je een traject controle zou kunnen implementeren op een snelweg. Hierbij gaan we

er voor het gemak van uit dat er camera's boven de weg hangen die een event kunnen genereren waarin het nummerbord en tijd verwerkt zitten. Verderop op de snelweg hangen wederom zulke camera's. Doordat de afstand tussen de camera's bekend is kan de gemiddelde snelheid berekend worden. Indien de snelheid te hoog is geweest wordt er een bekeuring verstuurd. Indien de snelheid lager is dan de toegestane snelheid wordt er niets met de events gedaan en verdwijnen ze uit het geheugen.

Hoe ziet StreamInsight eruit?

StreamInsight is een pluggable framework dat bestaat uit een engine en een adapter framework. Door middel van dit adapter framework kunnen ontwikkelaars interfaces bouwen met een keur aan verschillende databronnen. Deze databronnen kunnen sensoren of speciale meetapparatuur zijn, maar ook een web server of Windows NT-Events (ETW events), of zelfs een database. Hierop kom ik later nog terug.

Alle data in de StreamInsight Server bestaat in de vorm van streams. Iedere stream is in principe zonder einde, en verandert in de tijd. Stel dat je een digitale thermometer aansluit dan zal de temperatuur aan verandering onderhevig zijn naarmate de tijd verstrekt. De hoeveelheid tijd die tussen de events mag variëren. In figuur1 valt te zien dat er input en output adapters zijn. De output adapters kunnen bijvoorbeeld geaggregeerde- of gefilterde data wegschrijven naar een SQL Server tabel, een email of SMS versturen, een randapparaat aansturen of een bericht in een MSMQ-Queue wegschrijven. Als je het vanuit C# kunt aansturen dan kun je er een adapter voor bouwen. De engine zelf zorgt voor het continue verwerken van de events die de engine in gestreamed worden. De Query engine stuurt de resultaten van de verwerkte events naar de output adapter(s).

Input en output adapters

Een input adapter ontvangt event streams van een externe bron. Dit kan een socket zijn, een database table, sensoren etc. De input adapter ontvangt de events en vertaalt deze naar een structuur die door de StreamInsight server verwerkt kan worden. Wanneer de event source slechts één type event (structuur) produceert dan kan er een typed adapter gemaakt worden. In dit geval is de structuur van het event altijd identiek en bevat dezelfde velden. Deze velden zijn tevoren bekend en hoeven niet tijdens runtime uitgezocht te worden.

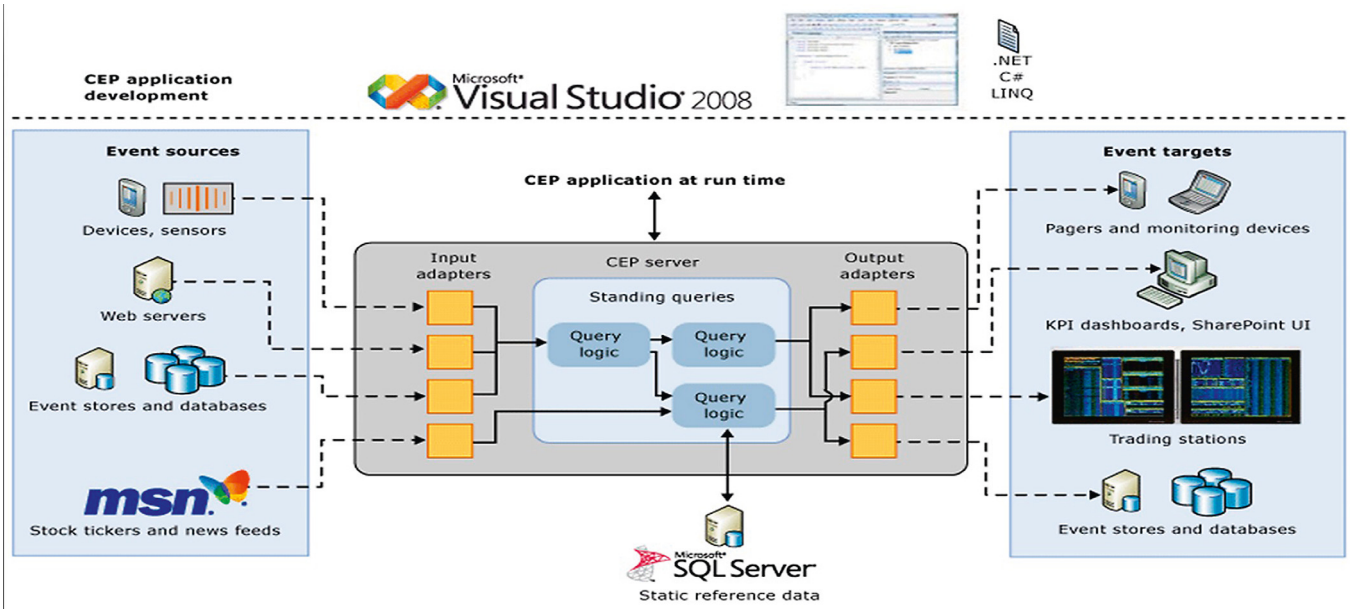
Als de event source verschillende typen events produceert die variëren kwa structuur dan moet er een untyped adapter geïmplementeerd worden. De velden van het event worden gedefinieerd tijdens de query bind time in de adapter. Deze variabele velden kunnen voorkomen waarbij er afhankelijk van de inhoud van het event velden bijkomen of weggaan. Dit kan ook het geval zijn indien er een adapter van het type 'SQL Server' gemaakt zou worden waarbij de velden de event structuur van de tabel hebben. Untyped adapters geven meer flexibiliteit ten opzichte van typed adapters.
Een output adapter verstuurt gegevens naar externe devices. Het programmeren en ontwerpen van een output adapter is identiek aan dat van een input adapter.

Events

De data in een stream bestaat uit events. Deze events bevatten de data die door de StreamInsight server verwerkt wordt. Ieder event bestaat uit twee onderdelen. Een header met hierin de metadata waarin het event soort wordt gedefinieerd en een of meerdere timestamps voor het event. De timestamps zijn afkomstig van de event sources. Het tweede deel is de Payload. Dit is een .NET datastruktuur met de uiteindelijke event data (bijvoorbeeld snelheid, liters water verbruikt, stroom etc.).

Queries en Query templates

Queries en Query templates zijn de bouwstenen voor de developer om een event processing applicatie mee te bouwen. De (continue) datastream wordt door deze queries bewerkt en doorgestuurd naar de output adapter. Een query template wordt geschreven door een combinatie van LINQ en C#. Met LINQ kunnen expressies geschreven worden die deze bewerkingen doen. LINQ ondersteunt composition zodat het resultaat van een expressie weer gebruikt kan worden als input voor een volgend bouwsteen. Doordat LINQ in .NET is ingebouwd is het een zeer natuurlijke manier om bewerkingen op data te doen.



FIGUUR 1

MET STREAMINSIGHT KUNNEN DE EVENTS VAN DE CAMERA'S BEDUIDEND SNELLER WORDEN AFGEWIKKELD DAN HET VERKEER...

De StreamInsight library bevat een generic type CepStream<T> waarop een aantal bewerkingen gedaan kunnen worden. De volgende query bewerkingen worden ondersteund door de StreamInsight Server.

Projection

Een projectie onstaat als er gebruik gemaakt wordt van clausule in LINQ. In de SELECT clausule kunnen ook berekening (bijvoorbeeld conversies) plaatsvinden. In code ziet dit er als volgt uit:

```
public class Foto
{
  public String kenteken ;
  public DateTime tijd;
}

var q = from c in VoertuigStream
          select new {kenteken = c.kenteken, tijd = c.tijd};
```

```
Filtering
Een filter werkt net zoals een filter dat met een where clause in
een SQL query. Er kan gefilterd worden op waarden of expressies.
public class Foto
{
  public String kenteken ;
  public double snelheid;
}

var q = from c in OutputStream
    where c.snelheid > 84
          select c;
```

Het resultaat is een stream met hierin alleen voertuigen die harder dan 84km per uur gereden hebben op het traject. Deze stream kan naar een output adapter gestuurd worden die een acceptgiro verstuurd.

Grouping

Door een group and apply te gebruiken kunnen bewerkingen gedaan worden over groepen van events. Dit kan handig zijn om gemiddelden of tellingen van events te doen. Door middel van de

apply clause wordt er een bewerking op de groep van events gedaan. De details van deze statements zijn te vinden op http://msdn.microsoft.com/en-us/library/ee391478.aspx

Aggregation

Een aggregatie is een bewerking die op een set werkt. Er wordt een bewerking zoals min, average, max of count uitgevoerd over een subset van de data. Agregaties werken alleen op een window, die in StreamInsight gedefinieerd zijn in het generic type CepWindowStream<T>.

Joining

Indien events in meerdere streams gecorreleerd moeten worden kan de join gebruikt worden. Dit doen we in het voorbeeld met de trajectcontrole door de events van camera 1 en camera2 te joinen op het nummerbord veld. De join is een bewerking op het generic type CepStream<T>. Dit ziet er als volgt uit:

```
public class Foto
{
  public String kenteken ;
  public DateTime tijd;
}

var result = from BeginEvent in FotoStream1
       join EindeEvent in FotoStream2
       on BeginEvent.kenteken equals EindeEvent.kenteken
       select new { kenteken = BeginEvent.kenteken,
             diff = DateDiff( DateInterval.Seconds, BeginEvent.
             tijd, EindeEvent.tijd };
```

Het resultaat van deze bewerking is een CepStream<T> met per kenteken de tijd in seconden die het voertuig er over heeft gedaan om het traject af te leggen. Deze stream kan weer kan dienen voor een volgende bewerking.

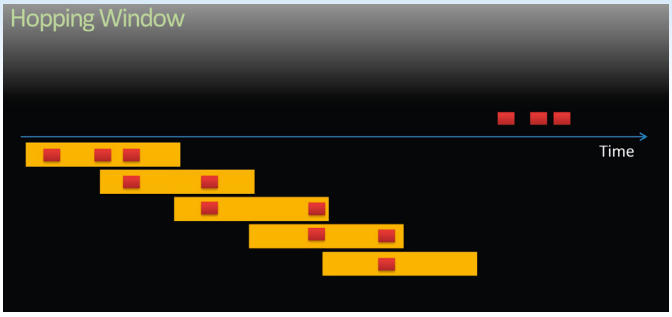
Unions

Stel dat er per rijbaan voor onze trajectcontrole een camera hangt die een opname van de auto's maakt. De fotostream van de camera's aan het begin van het traject kunnen tot één stream bewerkt worden door de Union bewerking. De Union is een bewerking op het generic type CepStream<T> en heeft als resultaat ook weer een CepStream<T>.

```
var allFotos = Rijbaan1.Union(Rijbaan2);
In het geval er meerde rijbanen zijn dan is het een kwestie van
een extra statement per rijbaan met weer een union. Dit ziet er
dan als volgt uit:
var tmpResult = Rijbaan1.Union(Rijbaan2);
var allFotos = tmpResult.Union(Rijbaan3);
```

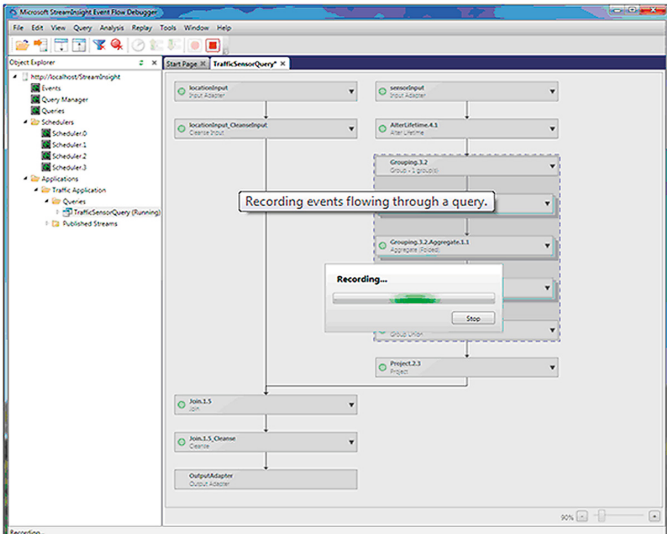
Windows

In een event processing applicatie bestaat er vaak een requirement om bewerkingen te doen op een subset van de data in een bepaalde periode. Voorbeelden hiervan zijn: "wat was het waterverbruik van de laatste 5 minuten", of "hoeveel aandelen zijn er verhandeld gegroepeerd per seconde over de laatste 4 uur". Hier komt ook de echte kracht van LINQ en StreamInsight naar voren. Op het eerste gezicht zijn dit queries die makkelijk met T-SQL gedaan kunnen worden. Maar wordt al wat lastiger als er bijvoorbeeld een moving average gevraagd wordt. Uiteraard kan dit ook met T-SQL gedaan worden maar de in-memory functionaliteit van StreamInsight maakt dit veel efficiënter te implementeren. Helemaal als er veel moeilijker windowing functionaliteit wordt gevraagd.



Hoe debug je een CEP applicatie?

Systemen die gebruik maken van StreamInsight zijn meestal systemen die heel erg veel events verwerken met complexe relaties tussen de events. Het vaststellen van de correctheid en het debuggen van deze events is vaak moeilijk. Dit komt mede door het tijdelijke aspect van deze events. Met het installeren van StreamInsight krijg je ook een debugger. Met de debugger kan de eventflow op een grafische wijze geïnspecteerd worden.



Samenvatting

StreamInsight opent nieuwe mogelijkheden voor SQL Server op het gebied van Continuous Event Processing applicaties. Het is in te zetten om sensordata, clickstream data, financiële data of bijvoorbeeld systeem events te verwerken in een low-latency, high-volume scenario. De StreamInsight server fungeert hierbij als de eerste verwerkingsstap om de hoeveelheid (ruwe)data die de database in dient te gaan helpt controleren.

Referenties

Het beginpunt voor StreamInsight op MSDN is http://msdn.microsoft.com/en-us/library/ee362541.aspx en het StreamInsight teamblog is te vinden op: http://sql-blog.com/blogs/stream_insight/default.aspx

De voorbeeld code voor de trajectcontrole in dit artikel kan via de blog van Paul van Wingerden gedownload worden op http://paulvanw.com.

Paul van Wingerden, is Developer Evangelist voor ISV's bij Microsoft