Progress Apama platform voor event stream processing

Balans tussen brute kracht en complexe acties

Robbert Hoeffnagel

Met zijn opmerkingen in het vorige nummer van Database Magazine over concurrenten als Apama, heeft Michael Stonebraker een interessante discussie op gang gebracht over event stream processing (ESP). Als de technische man achter StreamBase richt hij zich volledig op snelheid bij het verwerken van real-time binnenkomende data.

Bij ESP gaat het echter om meer, waarbij met name de vraag centraal staat of de te verwerken events uiterst simpel zijn of juist veel complexer. Het is dus maar de vraag of de claim dat StreamBase sneller zou zijn dan Progress Apama eigenlijk wel belangrijk is.

> Het is nodig de binnenkomende data zo snel mogelijk aan bewerkingen te onderwerpen

Apama werd eind 1999 opgericht door twee alumni van de Universiteit van Cambridge in Engeland. De tien jaar daarvoor hadden zij samengewerkt in het computer science-lab van deze universiteit. Hun idee was een oplossing te ontwikkelen voor een aantal real-time problemen waarmee mobiele telecommunicatiebedrijven te maken hebben. Daartoe was het nodig om in zeer korte tijd zeer grote hoeveelheden data te kunnen verwerken. Relationele databases kunnen dit soort snelheden niet aan, zodat nieuwe technologie ontwikkeld diende te worden.

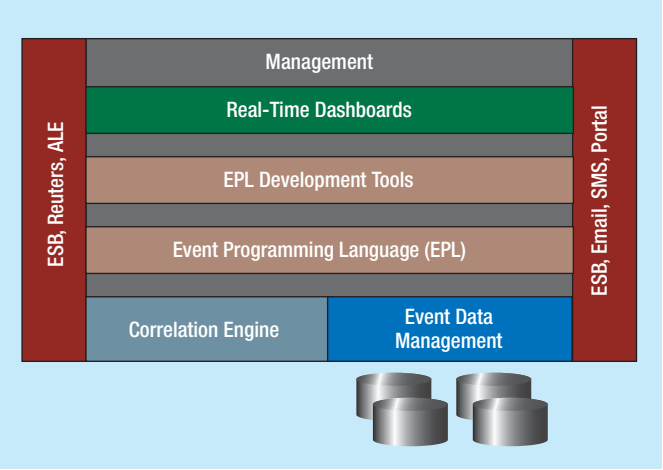
Brede doelgroep

Tot men zich realiseerde dat dit soort problemen niet alleen bij telco's voorkomt. Veel meer bedrijven zijn op weg naar wat Gartner wel 'the real-time enterprise' noemt en hebben behoefte aan oplossingen die het mogelijk maakt om in zeer korte tijd zeer grote hoeveelheden data te verwerken. Het klassieke model – eerst data opslaan, vervolgens indexen updaten en dan pas de bewerking doen – kost dan teveel tijd. Of zoals Stonebraker het in het artikel 'Sensorrevolutie stelt nieuwe eisen aan datamanagement' in DB/M 8 van 2006 uitdrukte: tien milliseconden is een eeuwigheid in elektronische handel.

Apama zat op het moment dat het eerste product op de markt kwam nog niet erg ruim in zijn geld. Vandaar dat een keuze werd gemaakt: alle aandacht ging uit naar event stream processing voor de financiële sector en dan met name gericht op wat wel genoemd wordt algoritmic trading. Inmiddels is Apama echter overgenomen door Progress Software en wordt hard gewerkt aan een verbreding van de gebruiksmogelijkheden. Voor Progress is de overname een logische stap, aangezien het bedrijf hiermee over een hiërarchie aan datamanagement-platformen beschikt: een klassieke relationele database (OpenEdge RDBMS), een 'in-memory' database (ObjectStore) en nu dus ook een platform voor het verwerken van real-time data (Apama).
Kenmerkend voor de eerder genoemde real-time enterprise is dat continu zeer grote hoeveelheden 'data feeds' worden gegenereerd die niet zozeer nodig zijn voor het opbouwen van historisch overzicht, maar die juist bedoeld zijn als stuurinformatie. Het is dus nodig de binnenkomende data zo snel mogelijk aan bewerkingen te onderwerpen, zodat direct op de hierdoor ontstane informatie kan worden gestuurd.

Alert engines

Conventionele methoden om data te verwerken in een real-time omgeving zijn hierbij niet toereikend en kunnen hooguit meekomen met de eisen van relatief kleinschalige toepassingen. Wie echter ondersteuning zoekt voor toepassingen waarbij de TCP/IP-stack per seconde – zeg – honderdduizend berichten doorstuurt, kan het bovendien wel vergeten dat een relationele oplossing dit nog aan kan.
Tot voor kort waren eigenlijk alleen de aloude *alert engines* in staat om enigszins in de buurt van de vereiste snelheden te komen. Probleem daarbij is echter dat deze technieken alleen geschikt zijn voor het bewaken van grenswaarden. Ze zijn eigenlijk alleen in staat om – toegegeven: razendsnel – vast te stellen of een bepaalde grenswaarde wordt overschreden. Er zit echter nauwelijks enige vorm van intelligentie in. Zo is het niet mogelijk om aan de hand van een reeks van meetwaarden te voorspellen of die grenswaarde 'binnenkort' overschreden gaat worden.
Een andere productcategorie die tot nu toe kanshebbend leek, bestaat uit de in-memory databases. Ook deze worden immers gepositioneerd als omgeving waarin de grote hoeveelheden data die gegenereerd worden bij het gebruik van bijvoorbeeld RFID-toepassingen kunnen worden opgevangen. Inderdaad zijn



Afbeelding 1. De opbouw van Progress Apama.

dit soort databases in staat om zeer grote hoeveelheden data te verwerken, ware het niet dat hierbij toch weer de klassieke volgorde wordt gehanteerd: eerst vastleggen, dan bewerken.

Event stream processing

In de real-time enterprise duurt dat te lang. Simpelweg meer hardware tegen het probleem aangooien – snelle multicore-processoren bijvoorbeeld – voldoet dan ook niet, aangezien de kosten hierdoor te zeer oplopen, terwijl de additionele overhead de schaalbaarheid zeker niet ten goede komt. Vandaar dat Apama – net als overigens een aantal andere aanbieders – voor een geheel andere aanpak heeft gekozen: eerst verwerken en dan pas opslaan (als dit laatste tenminste nodig is).

In afbeelding 1 zijn de elementen weergegeven waaruit Progress Apama is opgebouwd. Afbeelding 2 geeft een iets gedetailleerder beeld. Onderzoeksbureau Bloor Research wist deze manier van werken in een rapport aardig te typeren door te stellen dat bij event stream processing de data als het ware door de query stromen, terwijl bij conventionele methoden de query op de data wordt toegepast. Merk overigens op dat Apama niet bedoeld is als vervanger van bijvoorbeeld datawarehouses. Het is puur aanvullend gericht op real-time gegevensverwerking.

De in afbeelding 2 weergegeven smart blocks zijn voorgeprogrammeerde componenten die gebruikt kunnen worden om scripts of applicaties mee te ontwikkelen. Daarnaast is in de Apama Event Modeller sprake van scenario's. Dit zijn de query's plus de eventuele acties die uit het resultaat van een query volgen en die in de datastroom worden 'geplaatst'. De back testing- en de fault tolerance-modules zijn beide optioneel en dus niet direct noodzakelijk om met ESP aan de slag te kunnen. Hetzelfde geldt voor de niet in afbeelding 2 weergegeven Alert Engine, waarvan de functionaliteit duidelijk zal zijn.

Data toevoeren

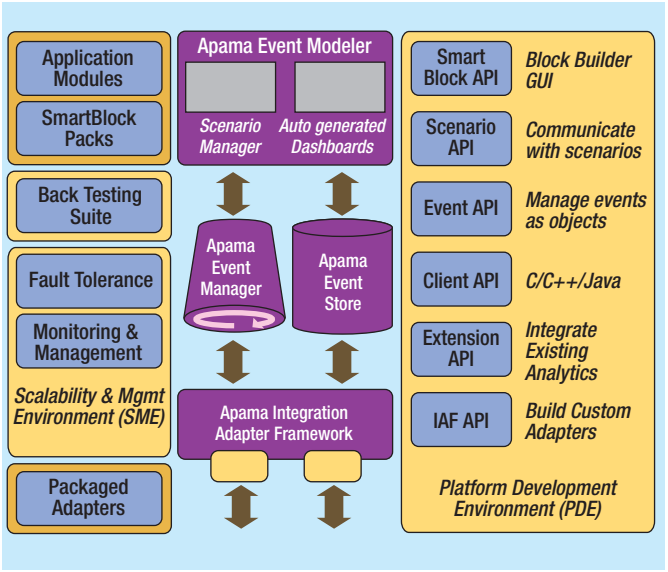
Verder wordt gebruik gemaakt van een zogeheten correlation engine. Het gaat hier om het mechanisme waarmee data in het systeem worden gebracht. Deze data kunnen uit tal van bronnen afkomstig zijn: een RFID- of ander sensornetwerk, het kan om een *live feed* met aandelenkoersen gaan, maar even zo goed om data uit een CRM- of ERP-pakket. De correlation engine brengt deze data in het systeem (zie afbeelding 3). Deze engine kan samenwerken met reguliere middleware maar kent ook API's om data uit specifieke bronnen via de engine direct in de EventStore (in feite de ObjectStore in-memory database) te brengen.

De correlation engine verwerkt alle data, maar kan bijvoorbeeld ook met delta's uit de voeten. Het zal in de praktijk namelijk regelmatig voorkomen dat een en dezelfde query ('monitoring scenario' heet dat formeel bij Apama) keer op keer wordt toegepast. In dat geval is het handiger – en sneller – om een query opnieuw te berekenen op basis van alleen die basisgegevens die zijn veranderd. Binnen een grotere hoeveelheid real-time data kunnen natuurlijk heel goed slechts enkele data-elementen zijn gewijzigd. Die hercalculatie lukt niet in een relationele omgeving. Een relationele database slaat nu eenmaal geen delta's op, terwijl SQL natuurlijk ook niet met dit soort functionaliteit in het achterhoofd is ontwikkeld. Bij Progress Apama kan dus wel met delta's worden gewerkt.

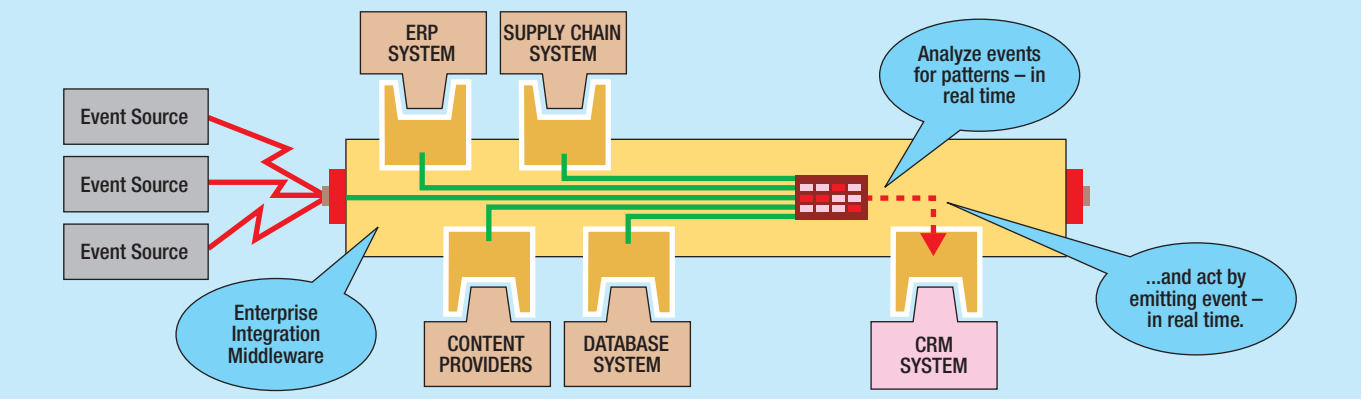
Het resultaat van een query kan worden teruggevoerd naar de oorspronkelijke applicatie. Dit gebeurt via een *publish/subscribe* methode dan wel ODBC. Daarnaast is de omgeving in staat om zelf ook resultaten weer te geven. Dat gebeurt dan via de module Dashboard Studio of via een standaard aanwezige voorziening die integratie met Microsoft Excel mogelijk maakt.

Query's bouwen

Waar het uiteindelijk natuurlijk allemaal om draait zijn de query's – de monitoring scenario's. Iedere query is opgebouwd uit een of meer monitors en de eerder genoemde smart blocks. Het verschil tussen beide is dat een monitor data herkent en doorsluist, terwijl een smart block tot een bewerking of het uitvoeren van een regel in staat is. De output van de een dient als input van de ander,



Afbeelding 2. De diverse modules waaruit Progress Apama is opgebouwd.



Afbeelding 3. Standaard middleware kan – samen met API's – worden gebruikt om data uit diverse bronnen in het systeem te brengen.

zodat stap-voor-stap een query kan worden opgebouwd. Smart blocks zijn bedoeld voor hergebruik en kunnen bovendien in geneste vorm worden toegepast.
Apama komt met een bibliotheek aan smart blocks, die echter aan de eigen wensen kunnen worden aangepast. Bij monitors gebeurt dit met een scripttaal (MonitorScript), een 'event processing language' die een aantal extra's heeft meegekregen. Denk naast de mogelijkheid om met delta's te werken, ook aan een voorziening om – via de Apama Package Manager – monitors te integreren met bestaande analytische routines. Ook is het mogelijk om patronen te definiëren waaraan de binnenkomende data moeten voldoen voordat deze naar een query worden doorgestuurd. Zo kunnen in een zeer vroeg stadium kaf en koren worden gescheiden, hetgeen heel wat nutteloze bewerkingen kan schelen.
Wie een query helemaal vanaf de basis wil opbouwen, gebruikt hiertoe de Event Modeller die deel uitmaakt van de op Eclipse gebaseerde Apama Developer Studio. Het gaat om een grafische werkomgeving waarin eenmaal aangemaakte monitors of smart blocks via *drag and drop*-handelingen in de juiste volgorde worden geplaatst. Het is mogelijk met versies te werken, zodat ook kan worden teruggevallen op eerdere versies van een query. Een belangrijk verschil tussen de Event Modeller en MonitorScript is dat de laatste uitsluitend voor ontwikkelaars is bedoeld, waar de eerste ook door business-medewerkers kan worden gebruikt.

State management

Tenslotte nog iets over 'event management'. Allereerst heeft Progress gezorgd voor een faciliteit die het mogelijk maakt om in het geval van een storing de status van events die gedurende een bepaalde periode worden verwerkt vast te leggen. Zeg maar: 'state management'. Hierdoor ontstaan mogelijkheden voor recovery, zodat het verwerkingsproces na een storing toch weer hervat kan worden. Dit gebeurt via zogeheten 'continuous availibility services' die als optie beschikbaar zijn.
Daarnaast is het mogelijk om events op te slaan. Bijvoorbeeld om deze in een later stadium alsnog te kunnen analyseren. Voor dit doel heeft Progress ObjectStore in Apama geïntegreerd, al heet deze hier dan EventStore. Events worden hierin vastgelegd als statische data. Op deze manier kan een set met historische data worden opgebouwd waaraan desnoods – via de eerder genoemde API's – ook data uit externe bronnen kunnen worden toegevoegd. Al deze data zijn dan beschikbaar om op iedere willekeurig moment aan een – zoals Progress het noemt – back test te worden onderworpen. Hiervoor is dan wel de eerder reeds genoemde optionele 'back test suite' nodig (zie afbeelding 1).

Op de shortlist

In een vorig jaar verschenen rapport toont onderzoeksbureau Bloor Research zich erg enthousiast over Progress Apama.
Het bureau gaat zelfs zover om vast te stellen dat iedereen die op zoek is naar een oplossing voor event stream processing Apama op zijn shortlist dient te zetten.

> Ook is het mogelijk om patronen te definiëren waaraan de binnenkomende data moeten voldoen

Inmiddels telt deze nog in de kinderschoenen staande markt al zo'n twaalf leveranciers. Snelheid van gegevensverwerking is een kenmerk van al deze concurrenten. De ene partij – en daar mogen we ongetwijfeld Stonebraker's StreamBase onder rekenen – richt zich daarbij met name op het behalen van een zo groot mogelijke snelheid. Andere aanbieders – en daar lijkt Progress met Apama een goed voorbeeld van – kijkt naar een iets breder toepassingsgebied waarbij met name de query plus opvolgende actie centraal staat.
Hoewel hierover nog nauwelijks gegevens te vinden zijn, zal de aanpak van de laatste groep ongetwijfeld een licht negatieve impact op de performance hebben. De vraag is of dat voor veel toepassingen werkelijk een probleem is. ESP is meer dan een alleen een alert-mechanisme. In een real-time enterprise speelt de op de query gebaseerde opvolgende actie immers een minstens even belangrijke rol.

Robbert Hoeffnagel is freelance journalist.