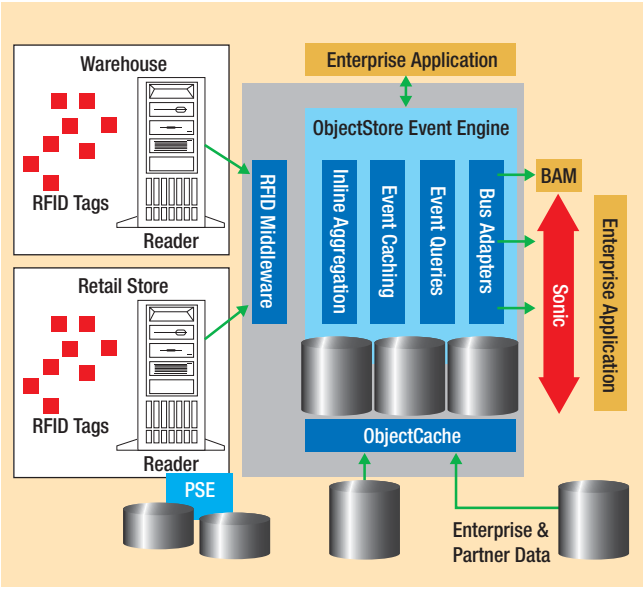
ObjectStore RFID Accelerator verwerkt 50.000 events per seconde

In-memory cache maakt real-time mogelijk

Robbert Hoeffnagel

Wat hebben bosbouw, RFID en database-technologie met elkaar te maken? Op het eerste gezicht niet zo vreselijk veel, ware het niet dat een bosbouwbedrijf als één van de eerste een pilot-project rond 'Radio Frequency Identification' uitvoert op basis van de nieuwe RFID Accelerator-programmatuur van ObjectStore. Deze dochteronderneming van Progress heeft een op een 'in-memory event cache' gebaseerde aanpak ontwikkeld die per seconde tot vijftigduizend 'events' kan verwerken. Mark Palmer van ObjectStore legt uit hoe deze technologie functioneert.

Cambium Forstbetriebe heeft een probleem: het bedrijf raakt jaarlijks tien tot vijftien procent van zijn producten kwijt. Dat is opmerkelijk aangezien deze Duitse firma zich bezighoudt met bosbouw en bomen nu eenmaal niet zo gemakkelijk zoek raken. Waar het productieproces in de bosbouw erg overzichtelijk lijkt – van winning via transport naar verwerking – gaat het administratief gezien om een complexe wereld. Er zijn bij houtwinning zeer veel en vooral veel kleine partijen betrokken, die allemaal een eigen administratie bijhouden en waar zelden sprake is van een vorm van standaardisatie die geautomatiseerde gegevensuitwisseling mogelijk maakt. Fouten bij het vastleggen van gegevens, maar ook vergissingen bij bijvoorbeeld het laden van vrachtwagens na winning of het leveren van verkeerde houtsoorten of -kwaliteiten aan klanten, betekenen dat een bosbouwbedrijf jaarlijks van iedere honderd bomen die het geveld heeft er zeven tot tien administratief kwijt raakt. De kostenpost die hierdoor ontstaat, bedraagt al gauw tweehonderd- tot vierhonderdduizend euro per jaar.



Afbeelding 1: Schematische weergave van de ObjectStore RFID Accelerator.

Tag

Om de kans op administratieve fouten terug te dringen, is Cambium in samenwerking met ObjectStore een pilotproject rond 'Radio Frequency Identification' (RFID) begonnen. Het idee is eigenlijk heel eenvoudig: voorzie iedere gevelde boom van een eigen 'tag' en lees bij iedere stap in het verwerkingsproces de productcode uit die in deze chip is vastgelegd. Op die manier staat bij iedere verwerkingsstap vast om welke boom het nu precies gaat. Door tevens een aantal basisgegevens vast te leggen – datum en locatie van vellen, boomsoort, kwaliteit en dergelijke – ontstaat al gauw iets dat op een database vol gegevens over bomen lijkt.

Dat was althans het idee. Men liep echter al snel tegen een probleem aan: hoe kan de grote hoeveelheid data die het uitlezen van de RFID-tags oplevert op een fatsoenlijke manier in een database worden ondergebracht? Voor de beeldvorming: ieder jaar wordt in Duitsland circa vijftig miljoen kubieke meter hout gewonnen. Dat betekent dat het gaat om vele tienduizenden bomen en evenzoveel 'tags' die tijdens hun tocht door de logistieke keten een groot aantal keer uitgelezen zullen worden. De ERP-leverancier van Cambium – het softwarebedrijf DABAC – stelde daarop voor om gebruik te maken van RFID Accelerator, een onlangs

door ObjectStore geïntroduceerd product dat is gebaseerd op een 'in-memory database', zie afbeelding 1.

In-memory data cache

"De ObjectStore RFID Accelerator is bij mijn weten de eerste 'in-memory event data cache' voor RFID-toepassingen," zegt Mark Palmer. Hij is als vice president verantwoordelijk voor de marketing bij ObjectStore. "Het product is gebaseerd op onze Event Engine. Deze is bedoeld voor het vastleggen, beheren en bewerken van gegevens die beschikbaar komen bij zogeheten 'streaming events'. Het product voldoet aan de EPCGlobal Tag Data Standard, een belangrijke norm om op het gebied van elektronische productcodes tot uniformering te komen. Dat wil onder andere zeggen dat ons product 'out of the box' over een EPC-schema beschikt."

ObjectStore is een dochteronderneming van Progress Software en richt zich op het ontwikkelen van producten die vallen onder de noemer 'real time data management'. Daarmee richt het bedrijf zich op een zeer specifiek segment binnen de markt: in-memory databases. Net als de andere producten van ObjectStore is de Event Engine gebaseerd op de zogenaamde 'Cache-Forward Architecture' (CFA) van het bedrijf. Deze aanpak voorkomt de 'latency', zo vertelt Palmer, die bij traditionele database-architecturen wordt veroorzaakt doordat 'requests' voor data worden uitgevoerd door een database server die – soms zelfs fysiek – op afstand is geplaatst van de applicatie. Voeg daarbij de tijd die verloren gaat met het omzetten van de data van een relationele structuur naar een structuur waarmee de applicatie uit de voeten kan en het is wat Palmer betreft duidelijk dat traditionele database-architecturen niet geschikt zijn voor een real-time toepassing als RFID.

> Een pipeline stage wordt geïmplementeerd in de vorm van een C++ 'functor'

"Bij CFA maken we echter gebruik van lokaal toegankelijke 'in-memory caches'. Hierdoor wordt voorkomen dat gewerkt moet worden met traditionele requests voor data. Daarnaast maakt CFA transacties tegen lokale caches mogelijk waarbij de integriteit van de data is gegarandeerd. Hierdoor weten we zeker dat iedere gedistribueerde 'cache instance' consistent is met de andere caches en de persistente gegevens op de server. Deze consistentie wordt afgedwongen via een mechanisme dat we 'Callback Locking' noemen. Hierbij worden 'locks' met de data in het geheugen geplaatst dat voor de applicatie is gereserveerd. De cache kan hierdoor een reeks van transacties tegen deze data uitvoeren, zonder dat er per transactie of een groep van transacties

Foto Harry Otto

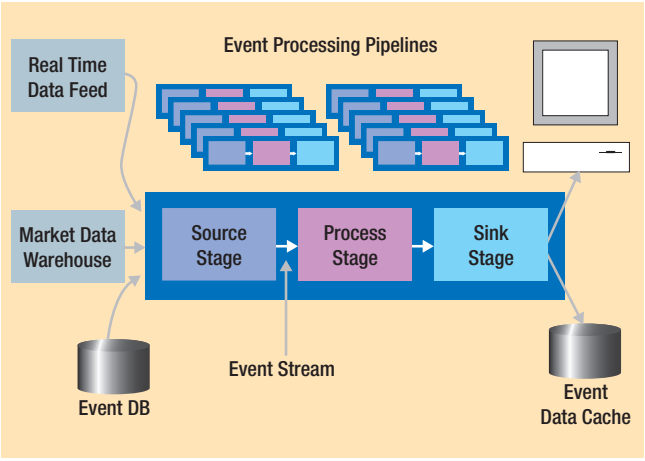
Mark Palmer, vice president marketing van ObjectStore.

data uit de cache van de server behoeven te worden opgevraagd. Ook is het niet nodig om opnieuw rechten aan te vragen voor het uitvoeren van opeenvolgende transacties. Alleen als in een cache geplaatste data door een ander proces wordt gewijzigd, zal de cache een nieuwe kopie van de benodigde data moeten ophalen."

Virtual memory mapping

Zou dit mechanisme niet worden toegepast, vertelt Palmer, dan vereisen transacties tegen de data in een cache, dat deze data na het afronden van de transactie eerst worden terug geschreven naar de server, waarna opnieuw 'locks' voor de daarop volgende transacties zullen moeten worden gerealiseerd. "Wij noemen dat ook wel 'repetitive lock acquisition'. Daarmee hebben we naar onze mening de achilleshiel van veel gedistribueerde architecturen kunnen lokaliseren. Bij dit soort architecturen vergen de veelvuldig optredende caching-operaties namelijk niet alleen een aanzienlijke hoeveelheid verkeer over het netwerk, maar veroorzaken zij tevens een stevige overhead ten aanzien van het beheer van de data. Met als resultaat dat de 'subsecond' responstijden die we bij realtime-toepassingen nodig hebben niet meer mogelijk zijn."

De Event Engine in de RFID Accelerator maakt verder gebruik van een zogeheten 'Virtual Memory Mapping'-architectuur (VMMA). Hierbij worden persistente data op een vergelijkbare wijze beheerd zoals hoe een besturingssysteem om gaat met 'virtual memory' en 'paging'. "Dat wil zeggen dat data-objecten op een dusdanige manier in het virtueel geheugen worden geplaatst ('gemapped') dat deze binnen het cache waarin zij zijn geplaatst, kunnen worden benaderd met de snelheden die we van 'in-memory' operaties zijn gewend.
Eén van de belangrijkste aspecten die het prestatieniveau van een applicatie bepalen, is de snelheid waarmee het zogeheten 'pointer



Afbeelding 2: De opbouw van een pipeline voor het verwerken van events.

dereferencing' plaatsvindt. In een optimale situatie is hiervoor slechts één instructie nodig. Bij veel architecturen lukt dat echter niet en zijn meerdere instructies nodig. Bij de aanpak die wij volgen vindt een automatische detectie van referenties naar data binnen de applicatie plaats. Bevinden de data zich niet in het geheugen, dan vindt een automatische verplaatsing van de 'pages' uit de lokale cache plaats, waarbij gebruik wordt gemaakt van de 'virtual memory' hardware van de CPU zelf. Zijn de data niet in de lokale virtuele cache aanwezig, dan ontdekt de hardware zelf deze fout en herstelt deze door middel van een 'call' die door CFA wordt herkend als een geheugenfout die wordt opgelost door de benodigde data uit het geheugen van de server op te vragen."

De combinatie van CFA en VMMA maakt het, zo stelt Palmer vast, niet alleen mogelijk dat optimaal gebruik wordt gemaakt van de mogelijkheden van de onderliggende hardware, maar betekent ook dat het beheer van persistente gegevens geheel automatisch plaatsvindt, zonder dat hier expliciete actie van de applicatie voor nodig is. "Dat wil zeggen dat de applicatie-ontwikkelaar zich dus geen zorgen hoeft te maken over de manier waarop zijn applicatie toegang tot data kan verkrijgen en hoe data na het afronden van de transactie verder beheerd dienen te worden. Tegelijkertijd is sprake van een prestatieniveau dat – doordat geheel 'in memory' kan worden gewerkt – ruimschoots voldoet aan de eisen die gesteld worden bij toepassingen als het 'in real time' verwerken van RFID-data."

Informatie over events

De basis voor de RFID Accelerator wordt dus gevormd door de op de hiervoor beschreven manier functionerende Event Engine. Hoe gaat dit product nu om met de data die bij het door scanners uitlezen van de RFID-tags richting deze database stroomt? Hierbij spelen twee begrippen een hoofdrol: 'events' en 'pipelines', zie afbeelding 2. Een event wordt in dit geval gedefinieerd als een gebeurtenis die tijdens een operatie of een activiteit is gemeten of waargenomen. Het gaat bovendien om persistente gebeurtenissen. Met andere woorden, informatie over deze events kan in een database worden vastgelegd.
Van ieder persistent event zijn drie typen van informatie bekend, vertelt Palmer. Dat is allereerst een 'category'. "Hierdoor kan een gebeurtenis samen met andere gelijksoortige events worden gegroepeerd. Dat maakt het mogelijk om in een later stadium groepen van gebeurtenissen te selecteren voor analyse of verwerking. Een voorbeeld van een categorie is de identificatie (ID) van een klant."
Het tweede type informatie heeft betrekking op een zogeheten 'ordering value'. Deze maakt het mogelijk om een gebeurtenis binnen een categorie te positioneren ten opzichte van andere events. Over het algemeen worden gebeurtenissen ten opzichte van elkaar geplaatst aan de hand van de factor tijd, die wordt uitgedrukt in milliseconden.
Als derde type informatie tenslotte dienen van een persistent event de nodige door de gebruiker gedefinieerde data beschikbaar te zijn. Denk bijvoorbeeld aan een getal dat een hoeveelheid aangeeft.

Pipelines en stages

Dan is er sprake van 'pipelines'. Applicaties die met de RFID Accelerator samenwerken kunnen een reeks van pipelines toepassen om de Event Engine-operaties te laten uitvoeren. In afbeelding 2 is weergegeven hoe een event door een uit drie delen of 'stages' bestaande pipeline stroomt. Events worden van de ene 'stage' naar de andere overgebracht door middel van een zogeheten 'event stream', waarbij een event stream die een event naar een stage brengt een 'input stream' wordt genoemd, terwijl een uitgaande event stream een 'output stream' heet.

> Consistentie wordt afgedwongen via een mechanisme dat 'Callback Locking' genoemd wordt

In het schema in afbeelding 2 bestaat de pipeline uit drie gedeelten. In de zogeheten 'producer stage' bestaat geen input stream, maar produceert de pipeline zelf events die het vervolgens in de output stream plaatst. Dit gebeurt op basis van externe events. Dit kunnen 'real-time feeds' zijn. De pipeline kan echter ook reeds vastgelegde events ophalen uit een database, uit een file of uit een andersoortige container. Bij het terughalen van een event uit een database kan de pipeline in de producer stage bovendien filters toepassen, zodat alleen events die in een bepaalde categorie vallen of die binnen een bepaalde tijdsinterval liggen, in de output stream terecht komen.
De 'processor stage' kent zowel een input stream als een output stream. De input stream bestaat uit events die zijn verwerkt door

de producer stage, de output stream bestaat uit events die het resultaat zijn van de verwerking die in deze fase plaatsvindt en die overgebracht dienen te worden naar de volgende fase, de 'consumer stage'. Welke bewerkingen in de processor-fase precies plaatsvinden, hangt sterk af van wat de betrokken applicatie met de events wil doen. ObjectStore noemt in de documentatie onder andere het verzamelen en rapporteren van statistieken die het resultaat zijn van analyses van events. De resultaten van deze analyses worden vervolgens in de output stream geplaatst. Een ander voorbeeld betreft het omzetten van een event in een ander event, wat bijvoorbeeld nodig kan zijn als de data van een event gecorrigeerd dienen te worden.
De laatste fase van dit voorbeeld van een pipeline wordt de 'consumer stage' genoemd. In dit deel wordt het resultaat van de eerdere fasen de pipeline uitgebracht. Als de pipeline onderdeel is van een applicatie waarop meerdere gebruikers zijn geabonneerd ('subscription'), dan zorgt deze stage bijvoorbeeld voor de distributie van de resultaten naar alle abonnees. Bovendien vindt in deze stage alle verwerking plaats die eventueel nodig is om dit te realiseren. De consumer stage zorgt er tevens voor dat de pipeline wordt beëindigd.

Foto Harry Otto

Mark Palmer: "Traditionele database-architecturen zijn niet geschikt voor een real-time toepassing als RFID".

Ieder pipeline heeft in ieder geval een producer stage en een consumer stage nodig en kan één of meer processor stages omvatten. Een pipeline stage wordt geïmplementeerd in de vorm van een C++ 'functor'. Functors zijn C++ classes en kunnen van 'arguments' worden voorzien die helpen bij het verwerken van events. De producer stage en de consumer stage tellen ieder één functor, een processor stage kan meerdere functors omvatten.

EPCIS-standaard

"Op basis van deze architectuur zijn we erin geslaagd de EPC Information Service (EPCIS) standaard van EPCGlobal te implementeren," zegt Palmer. "Dat we deze standaard ondersteunen, betekent onder andere dat we een reeks van adapters meeleveren waarmee RFID-gegevens uit een hele reeks van bronnen kunnen worden verzameld. Hierbij ondersteunen we 'Application Level Event' (ALE) 1.0-specificatie en leveren we een query adapter die het mogelijk maakt om via het 'Simple Object Access Control' (SOAP) model de event-geschiedenis te bevragen. Query's worden hierbij gespecificeerd als in XML geformatteerde 'Event History Reports' ofwel EHreports." Overigens is ALE 1.0 nog niet formeel vastgesteld. De status van deze specificatie is vooralsnog dan ook 'draft'.

Daarnaast levert ObjectStore, zoals te zien in afbeelding 1, de nodige RFID-middleware mee. "Daar gebruiken we RFTagAware van ConnecTerra voor. Deze programmatuur is in staat om op basis van ALE 1.0 'tag readers' te beheren en data te verzamelen. Dat betekent dat we kunnen samenwerken met vrijwel alle populaire readers, printers en tag-formaten."
Om de stroom gegevens over events te kunnen verbinden met bedrijfsapplicaties is tenslotte nog een koppelingsmechanisme nodig. Conceptueel werkt ObjectStore hiervoor met een 'messaging bus', die concreet is ingevuld met het van zusterbedrijf Sonic afkomstige Sonic ESB.

Nog geen 100 bits

Hoewel het per uit te lezen productcode om nog geen 100 bits gaat, spreekt men ook bij een pilotproject als Cambrium Forstbetriebe vanwege het grote aantal bomen van zeer grote datavolumes. Nadat iedere gevelde boom van een tag is voorzien, wordt met draagbare readers de productcode van individuele bomen bij ieder stap in de logistieke keten gelezen. De gelezen data worden via een GSM-verbinding doorgeseind naar de centraal opgestelde RFID Accelerator-module die DABAC aan het ERP-pakket van het houtvestbedrijf heeft toegevoegd.
Het bedrijf heeft redelijk hoge verwachtingen van de voordelen die deze manier van werken uiteindelijk zal opleveren.
Palmer: "Het management van Cambrium Forstbetriebe verwacht dat het verlies aan bomen met circa 70 procent zal kunnen worden teruggedrongen. Bovendien denkt men tot een halvering van de handling-kosten te kunnen komen."

Robbert Hoeffnagel is freelance journalist.