Schaalbaarheid, multi-tier-omgevingen en de comeback van TP-monitoren

Ing. R.J.A. Stouthart

TP-monitoren zijn terug van weggeweest. Wanneer traditionele client-serveromgevingen onvoldoende mogelijkheden bieden om grootschalige én robuuste IT-infrastructuren te realiseren, kan de inzet van een TP-monitor uitkomst bieden. In dit artikel wordt beschreven wat een TP-monitor is en welke voor- en nadelen deze middleware heeft voor het ontwerpen, bouwen en exploiteren van grootschalige IT-infrastructuren.

Inleiding

Transaction Processing-monitoren (TP-monitoren) staan weer in het centrum van de belangstelling. Jarenlang domineerden zij de mainframes, maar zij werden naar de achtergrond verdrongen door de opkomst van de client-servertechnologie en midrangeomgevingen. Door de opkomst van intranetten en het Internet, applicatiepartitionering en componentgebaseerd ontwikkelen, is de noodzaak van TP-monitoren weer toegenomen en zijn TP-monitoren bezig met een comeback.

Bij het bouwen van informatiesystemen wordt momenteel voornamelijk gebruikgemaakt van traditionele client-serverconcepten. Deze architectuur blijkt in de praktijk een aantal beperkingen met zich mee te brengen. Deze beperkingen liggen op het vlak van de schaalbaarheid en hebben tot gevolg dat bij grote gebruikerspopulaties de beschikbaarheid en betrouwbaarheid van de IT-infrastructuur terugloopt.

Dit artikel poogt een overzicht te geven van de mogelijke argumenten voor een overgang van een traditionele two-tier client-serverarchitectuur naar een multi-tierarchitectuur op basis van middleware zoals TP-monitoren. Hoge eisen aan de schaalbaarheid en de beschikbaarheid van een omgeving vormen de katalysator voor deze overgang.

Als eerste worden de architectonische (on)mogelijkheden beschreven van traditionele monolithische en client-serverarchitecturen voor het ondersteunen van grootschalige gebruikerspopulaties. Daarna worden de argumenten beschreven voor een overgang naar een multi-tierarchitectuur op basis van TP-monitoren. Vervolgens wordt aan de hand van een vergelijking met de traditionele transactiebesturingsmechanismen binnen een typische two-tieromgeving een wensenlijst opgesteld voor een TP-monitor. Aansluitend wordt de functionaliteit (en voor de liefhebber tevens de architectuur) van TP-monitoren uitgediept. Afsluitend wordt een beeld geschetst van recente ontwikkelingen en toekomstverwachtingen op het terrein van multi-tieromgevingen en TP-monitoren. Het artikel wordt afgesloten met een samenvatting en een literatuuroverzicht.

Het ontstaan van TP-monitoren

Deze paragraaf beschrijft het ontstaan van TP-monitoren als gevolg van een schaalbaarheidsprobleem en het optreden hiervan binnen monolithische en client-server-architecturen.

Het probleem

Binnen een transactiegeoriënteerde omgeving maken gebruikers verbindingen met applicatie- en databaseservers. Een architectuur die gebruikers en hun workload niet op een efficiënte manier ondersteunt, zal onvermijdelijk leiden tot performance-, beschikbaarheids- en beheerbaarheidsproblemen. In de loop van de tijd wordt het steeds moeilijker een groeiend aantal gebruikers en daarmee een groeiend aantal transacties te verwerken, waardoor de beperkingen en gevolgen van het gebrek aan schaalbaarheid duidelijk worden. Nieuwe en krachtiger hard- en software binnen dezelfde architectuur rekt grenzen op maar kan deze niet overschrijden.

Een architectuur wordt zelden onderkend als oorzaak van schaalbaarheidsproblemen. Performanceproblemen worden meestal geweten aan de snelheid van het platform, de bandbreedte van het netwerk, de efficiency van het besturingssysteem of de congestie[1] binnen het databasemanagementsysteem. Performanceproblemen kunnen ook een gevolg zijn van duizenden on-linegebruikers die een meer dan wenselijk beslag leggen op CPU-tijd, geheugen en andere resources. De hieruit voortvloeiende beschikbaarheidsproblemen worden dan vaak geweten aan falende netwerkverbindingen of gebrek aan bandbreedte, terwijl zou moeten worden geaccepteerd dat netwerkverbindingen kunnen én zullen falen en dat een systeemarchitectuur moet worden gebruikt met voldoende robuustheid om hiermee om te gaan.

Het monolithische model

Bij het monolithische model bedient een centrale server vele 'domme' terminals. Typische voorbeelden zijn mainframeomgevingen en het Internet. Voor iedere gebruikerssessie op een terminal is een dedicated proces actief op de server. In dit clientproces worden de businesslogica en de presentatielogica voor de terminal verzorgd. Daarnaast communiceert het clientproces via een eigen verbinding met het databasemanagementsysteem. De verschillende processen, hun onderliggende verbindingen en de verbinding van de terminal naar het corresponderende clientproces blijven gehandhaafd zolang de gebruikerssessie actief is, ongeacht intensiteit van het gebruik.

1) Samenloop, verstopping. Congestie treedt op wanneer processen moeten wachten op het vrijkomen van resources die door andere processen worden gebruikt en gedurende deze tijd zijn 'gelocked'.

Schaalbaarheidsproblemen binnen de monolithische omgeving

De monolithische architectuur kent een aantal problemen met het ondersteunen van grote gebruikerspopulaties. Deze architectuur vereist twee processen voor iedere terminal: één voor het client-terminalproces en één voor het client-hostproces (zie figuur 1). Vijfhonderd gebruikers zullen duizend processen opstarten (plus de achtergrond- en besturingssysteemprocessen), wat leidt tot een hoog geheugenverbruik en een hoge CPU-overhead voor het onderhouden en swappen[2] van alle processen.

Het aantal processen leidt tot interne congestie binnen het besturingssysteem en het databasemanagementsysteem. Dit leidt tot groeiende vertragingen wanneer het gebruikersaantal of hun activiteit toeneemt.

De congestie en de daaruit voortvloeiende vertragingen zijn het gevolg van het inefficiënte resourcegebruik. Voor iedere nieuwe connectie worden nieuwe kopieën van clientprocessen gemaakt. Deze nieuwe kopieën worden weinig gebruikt omdat de meeste gebruikers het merendeel van de tijd inactief zijn. De processen voor deze connecties worden inefficiënt gebruikt en leggen een onnodig beslag op het geheel van beschikbare geheugen- en CPU-resources. Dit principe kan worden vergeleken met het model van een telefooncentrale. Alle lijnen zijn aanwezig en staan open, ongeacht of iemand iets te melden heeft.

Een efficiëntere methode voor het verdelen van resources dient te voorzien in een mechanisme waarbij tijdelijk resources (geheugen en verwerkingstijd) worden toegewezen om een gebruikersaanvraag af te handelen. Na afloop worden de resources doorgegeven aan andere processen totdat weer een gebruikersaanvraag binnenkomt. Ook hier kan worden vergeleken met het model van een telefooncentrale. Alle lijnen zijn aanwezig maar alleen voor een gesprek zal een verbinding tussen twee aansluitingen worden geactiveerd door de centrale.

Zoals uit de inleiding blijkt is binnen mainframeomgevingen deze behoefte reeds lang geleden onderkend en ingevuld. CICS, een TP-monitor voor het MVS-besturingssysteem van IBM, is de bekendste en meest gebruikte TP-monitor binnen mainframeomgevingen.

Het client-servermodel

Het client-servermodel is een uitbreiding op het monolithische model waarbij twee afzonderlijke client- en server-softwarelagen worden ondersteund, ieder op een eigen platform. Het client-servermodel kwam eind jaren tachtig in zwang en luidde een tijdperk in van decentralisatie, downsizing, midrangesystemen en PC's.

Bij het client-servermodel is het clientproces verplaatst naar een intelligent werkstation zoals een PC, dat communiceert met servers door middel van eenvoudige (door de serverleverancier meegeleverde) middleware[3]. In figuur 2 is weergegeven hoe de afzonderlijke clients samenwerken met een centrale (database)server.

De businesslogica[4] wordt verdeeld over de client en de databaseserver (afhankelijk van de mogelijkheid van het databasemanagementsysteem om stored procedures[5] te ondersteunen). Wanneer businesslogica op de client aanwezig is, voorziet de middleware in een data-passing-protocol waarmee data kunnen worden overgehaald naar de client om daar te worden bewerkt. Wanneer businesslogica op de databaseserver aanwezig is, voorziet de middleware in een message-passingprotocol om aanroepen van procedures binnen de database mogelijk te maken. Net als in de monolithische architectuur blijven alle verbindingen gehandhaafd tijdens de sessie, ongeacht de intensiteit van het gebruik.

Schaalbaarheidsproblemen binnen een two-tier-omgeving

De two-tier client-serverarchitectuur komt tegemoet aan enige onderkende knelpunten binnen de monolithische omgeving en biedt daarmee een grotere schaalbaarheid. Deze architectuur introduceert echter een aantal andere problemen die de schaalbaarheid weer beperken.

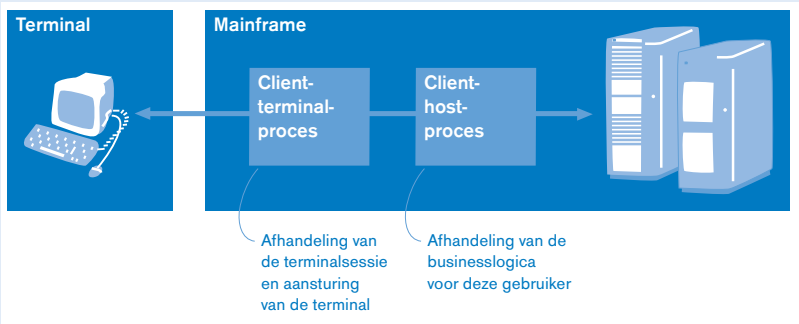
Client- en serverprocessen weten niet dat ze niet meer samen op dezelfde machine draaien. De gescheiden client-serverprocesen hebben geen weet van het feit dat ze worden gescheiden door een LAN of WAN en zijn daarom niet ingericht op het minimaliseren van netwerkverkeer (het aantal messages en/of data dat wordt uitgewisseld).
Bij het monolithische model is de transfertijd tussen processen op de dezelfde server zodanig klein dat de transfertijd verwaarloosbaar is; voor de transfertijd over een netwerk ligt dit anders. Een optimale architectuur dient dus netwerkverbindingen te onderkennen en het dataverkeer tussen de processen in aantal en omvang te minimaliseren.

Hoewel de two-tier client-serverarchitectuur oplossingen biedt voor een aantal problemen, blijven enkele problemen onopgelost en worden nieuwe geïntroduceerd.

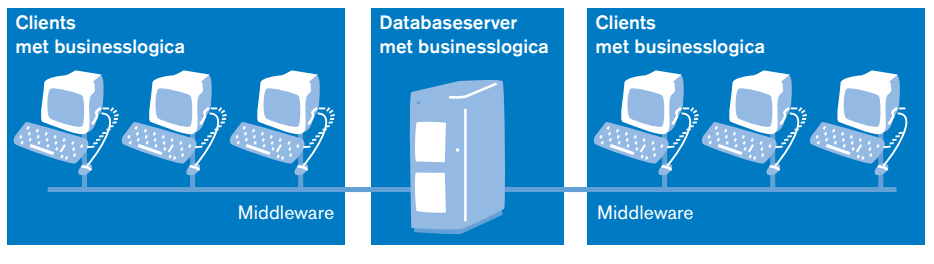
Andere connectivityproblemen

Een connectivityprobleem kan optreden als gevolg van het aantal gebruikers dat is geconnect aan een (database)server. Daarnaast kunnen connectivityproblemen optreden door het continue aan- en afloggen van gebruikers en de overhead die het opbouwen en afbreken van sessies met zich meebrengt. Dit connectivityprobleem treedt onder andere op bij het gebruik van connection-

2) Swappen is het in- en uitladen van gegevens uit het interne geheugen van een computersysteem. Swappen is een gevolg van een tekort aan interne-geheugenruimte om alle benodigde gegevens gelijktijdig te bevatten.
3) Middleware is een verzamelnaam van applicaties die fungeren als tussenlaag tussen de applicaties en het besturingssysteem. Middleware biedt applicatieondersteunende functionaliteiten zoals databasemanagement, message queueing, dynamic naming services en remote procedure calls (RPC).
4) De programmatische vastlegging van de bedrijfsregels.
5) Een stored procedure is een applicatie die is opgeslagen in een database en ook wordt uitgevoerd binnen de database. Databasemanagementsystemen die stored procedures ondersteunen, zijn naast een opslagplaats voor gegevens, tevens een soort applicatieserver waarbinnen applicaties kunnen draaien.



Figuur 1. Gebruikersprocessen binnen de monolithische omgeving.



Figuur 2.
Het client-server-model.

less protocollen. Bij connectionless protocollen wordt voor iedere actie een kortstondige sessie opgebouwd. De verbinding wordt telkens verbroken na een kleine hoeveelheid werk en kort daarna weer opgezet wanneer de gebruiker een volgende handeling verricht. Connectionless protocollen worden gebruikt binnen Internet- en intranetomgevingen en door de toenemende populariteit van deze omgevingen komt dit connectivityprobleem steeds vaker voor.

Binnen het Internet en intranetten wordt dit connectivityprobleem ondervangen door een nieuwe generatie webservers die alle aanvragen routeren via een (Web) Request Broker naar applicaties. Dit in tegenstelling tot de traditionele webservers op basis van CGI- en PERL-scripts. De Request Brokers handhaven een doorlopende verbinding met de applicatie en de applicaties handhaven op hun beurt een doorlopende verbinding met één of meer databases. De Request Brokers zijn erop ingesteld om gebruikerssessies te kunnen opzetten en afbreken met een maximale snelheid en een minimale overhead en bieden daarmee typische TP-monitorfaciliteiten.

Er zijn bedrijfstakken en bedrijfsprocessen die meer waarborgen vragen dan een traditionele client-serveromgeving kan bieden.

Clients sturen via de Request Brokers losse berichten (messages) aan de applicatieprocessen en ontvangen berichten die alleen gegevens bevatten die voldoen aan de vraag of bewerking. De heen- en weergaande stroom gegevens tussen de client en de databaseserver is nu vervangen door een staccato van kleine berichten. Deze, op berichten gebaseerde techniek, vormt de basis van de multi-tierarchitectuur die hierna zal worden toegelicht.

De overgang naar multi-tierarchitecturen

Two-tieromgevingen bieden steeds meer mogelijkheden om grotere aantallen gebruikers en transacties te ondersteunen. Toch zijn er bedrijfstakken en bedrijfsprocessen die meer waarborgen vragen qua performance, beschikbaarheid en beheerbaarheid dan een two-tieromgeving kan bieden. Deze paragraaf behandelt een voorbeeld van een omgeving waarin een multi-tierarchitectuur een goede oplossing biedt. Vervolgens wordt aan de hand van een vergelijking met de traditionele transactiebestu-ringsmechanismen binnen een typische two-tieromgeving een wensenlijst opgesteld voor een TP-monitor.

Een grootschalige omgeving

Behalve aan intranetten en het Internet kan bij een grootschalige omgeving bijvoorbeeld worden gedacht aan een bedrijfsketenbrede implementatie van een ERP-toepassing. Medewerkers over het hele land verrichten transacties, waarbij gebruik wordt gemaakt van verschillende resources. Gebruikers werken op vele verschillende typen werkstations en verwachten een goede performance en een beschikbaarheid van zeven maal vierentwintig uur. Hiervoor moeten een IT-architectuur en een informatiesysteem worden ontworpen die aan al deze wensen tegemoet komen en blijven komen gedurende de levensduur van het informatiesysteem.

Hardware gaat stuk, schijven lopen vol, locaties veranderen, nieuwe gebruikers worden toegevoegd en de businesslogica verandert. Tijdens al deze, op voorhand te verwachten, gebeurtenissen moeten het gewenste serviceniveau en de betrouwbaarheid van de gegevensverwerking zijn gewaarborgd. Multi-tierarchitecturen bieden hier een oplossing.

Multi-tierarchitecturen

Bij een verdere opsplitsing van de twee lagen binnen de two-tieromgeving ontstaat een nieuwe architectuur: de multi-tierarchitectuur. Hierbij ontstaat een laag voor de presentatie, een laag voor de bedrijfsregels en een laag voor de opslag van data. Binnen deze drie lagen (ook bekend als three-tierarchitectuur) is de laag van de bedrijfsregels een hiërarchie van elkaar diensten verlenende applicatiecomponenten. Hierdoor ontstaat een model met een bijna onbeperkte gelaagdheid. Dit model wordt aangeduid met de term multi-tier of n-tier.

De applicaties die de bedrijfsregels implementeren, kunnen worden gepartitioneerd tot afzonderlijke eenheden die afzonderlijk worden ontwikkeld. Door de mogelijkheden tot fijnmazige opsplitsing kunnen applicaties en ontwikkelteams zich wijden aan kerntaken en de resterende taken uitbesteden aan andere onderdelen binnen de multi-tierarchitectuur. Figuur 3 schetst een applicatie die is opgebouwd uit een groot aantal componenten. De pijlen symboliseren de TP-monitor die als middleware de verschillende componenten met elkaar verbindt. De afzonderlijke componenten kunnen zijn gedistribueerd over afzonderlijke computersystemen maar kunnen zich ook binnen één enkel computersysteem bevinden.

Om systemen te kunnen realiseren binnen midrangeomgevingen die kunnen voldoen aan beschikbaarheids-, performance- en schaalbaarheidseisen die vergelijkbaar zijn aan de eisen die gelden in mainframeomgevingen, is een architectonische overgang nodig. Deze overgang wordt mogelijk gemaakt door de inzet van een TP-monitor.

Ook zonder de inzet van een TP-monitor kunnen multitieromgevingen worden gecreëerd. Ontwikkelaars kunnen er ook voor kiezen om de afhandeling van de communicatie tussen de afzonderlijke applicatieonderdelen zelf te specificeren en te programmeren. Dit brengt een aanzienlijke ontwikkelinspanning met zich mee. Daarnaast zullen de functionele eisen aan de communicatiemogelijkheden en de betrouwbaarheid hiervan op een steeds hoger niveau komen te liggen. Hierdoor loopt men het reële risico dat men uiteindelijk zelf een TP-monitor heeft gebouwd.

TP-monitoren worden momenteel reeds volop ingezet maar verlenen hun diensten in anonimiteit en worden ook vaak niet als TP-monitor aangeduid. Het gaat hier om de TP-monitoren die door applicatie- en databaseleveranciers worden meegeleverd als onderdeel van een applicatie of database. De programmatuur die de communicatie verzorgt tussen de client en de applicatie of database is een TP-monitor. Deze TP-monitoren worden TP-lights genoemd vanwege de beperkte functionaliteit van deze TP-monitoren. Deze functionaliteit verschilt overigens per databaseleverancier en per uitgebrachte release. Een voorbeeld van een TP-monitor die door een databaseleverancier standaard wordt meegeleverd, is de MTS (Microsoft Transaction Server) van Microsoft. Dit product is oorspronkelijk als zelfstandige TP-monitor uitgebracht maar redde het niet als zelfstandig product. MTS wordt tegenwoordig standaard meegeleverd als component van Microsoft Back Office.

Een TP-monitor, zoals beschreven in dit artikel, wordt een TP-heavy genoemd. Een TP-heavy biedt, naast functionaliteiten voor databases, functionaliteiten voor bijvoorbeeld load balancing, fail-over en security management. Deze TP-monitoren zijn op zichzelf staande commerciële producten en worden aangeboden door leveranciers die in deze materie zijn gespecialiseerd.

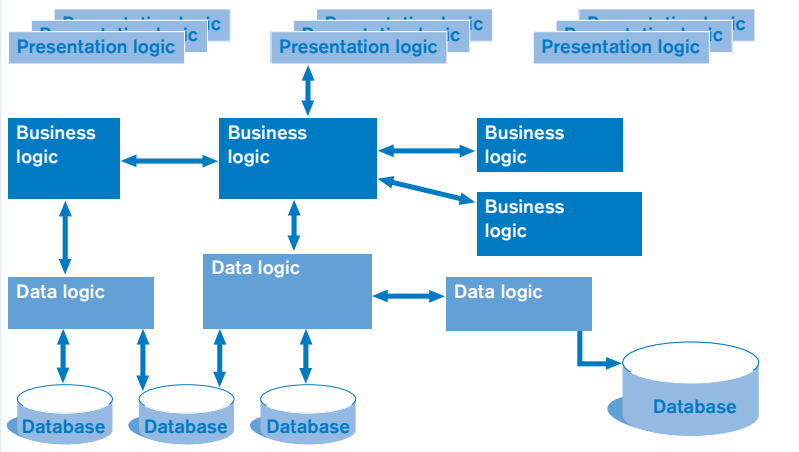
Eisen en wensen voor een TP-monitor

Een overzicht van tekortkomingen van two-tieromgevingen vormt tevens een eisen- en wensenlijst voor grootschalige omgevingen. Aan deze eisen en wensen kan worden voldaan door gebruik te maken van een bindmiddel dat de benodigde functionaliteit ter beschikking stelt. Een middlewareproduct dat deze functionaliteit biedt, vormt in wezen de implementatie van een TP-monitor.

Heterogene resources

Binnen een two-tieromgeving worden transacties meestal uitgevoerd tegen één soort databasemanagementsysteem.

Binnen een grootschalige omgeving moeten binnen één enkele transactie vaak meerdere (mogelijk verschillend soortige) datasources worden benaderd en gemanipuleerd. Deze datasources hoeven geen databasemanagementsysteem te zijn maar kunnen ook ASCII-bestanden of queues zijn.



Figuur 3. Multi-tierarchitectuur.

Asynchrone verwerking van taken

Binnen two-tieromgevingen wordt normaliter een server aangeroepen en gewacht tot de server de handeling heeft afgerond. Hierbij wordt de werking van de clientapplicatie zolang geblokkeerd tot de server een antwoord heeft gegeven.

Binnen een grootschalige omgeving dienen nog andere mogelijkheden te bestaan voor het creëren van complexere conversatiemodellen, zoals blokkeren (de applicatie wacht na iedere aanroep op een antwoord), asynchrone uitvoering (de applicatie verstrekt een opdracht en gaat verder met haar eigen taken) en aanroepen op basis van het optreden van een event (de server bericht de clientapplicatie van een gebeurtenis).

Ontwikkeltalen

Stored procedures binnen databases worden geschreven in leverancierspecifieke talen waardoor een afhankelijkheid ontstaat van een bepaald type database. Dit is één van de redenen waarom leveranciers van standaardpakketten zo min mogelijk businesslogica in een database programmeren.

Databaseontwikkeltalen draaien meestal alleen binnen de database en niet op de client waardoor een ontwikkelaar meerdere talen moet beheersen of gescheiden teams ontstaan voor client- en databaseserverprogrammering. Bovendien richten dergelijke talen zich op bepaalde soorten applicaties; andersoortige applicaties dienen door middel van een ander ontwikkelhulpmiddel en -taal te worden gerealiseerd.

Binnen een grootschalige omgeving moet businesslogica in willekeurige en meerdere talen kunnen worden geschreven en voor meerdere platformen. Daarnaast moeten globale transacties kunnen worden gedefinieerd die meerdere programma's kunnen omspannen. Deze globale transacties committen of rollbacken als een eenheid om daarmee de integriteit van een transactie te waarborgen.

Modulaire ontwikkeling
Binnen two-tieromgevingen wordt businesslogica ofwel in de clientprogrammatuur ofwel in de databaseprogrammatuur geplaatst; een verdere partitionering is niet mogelijk. Latere herpartitionering gaat, doordat de clientontwikkeltaal en de serverontwikkeltaal in veel gevallen verschillend zijn, gepaard met een hernieuwde ontwikkelinspanning. Bij de overgang van monolithische omgevingen naar client-serveromgevingen dreigt daarmee het ontstaan van twee nieuwe monolieten.

Binnen grootschalige omgevingen is het wenselijker om componentgeoriënteerd te werken waarbij GUI-specialisten presentatielogica ontwikkelen, functionele specialisten businesslogica ontwikkelen en dataexperts databases. Wanneer de interfaces tussen de afzonderlijke lagen zijn gedefinieerd, dan kunnen deze ontwikkelgroepen onafhankelijk van elkaar werken.

Nieuwe functionaliteit kan worden aangebracht door nieuwbouw of aanpassingen. Doordat services echter relatief autonome objecten zijn die alleen via voorgedefinieerde interfaces communiceren, wordt het eenvoudiger om services uit te wisselen met nieuwe of verbeterde services. Deze services kunnen zelf zijn ontwikkeld of worden ingekocht van derden. Hiermee wordt het mogelijk componentgeoriënteerd te gaan werken en worden de beheersings- en afbreukrisico's van omvangrijke projecten ondervangen.

TP-monitoren zijn 'proven technology' maar nog geen 'common technology'.

Tuning van apparatuur
Binnen een two-tieromgeving zullen beide delen van de applicatie programmacode kunnen bevatten. Hierdoor is het moeilijk een client of een databaseserver te optimaliseren en zullen compromissen moeten worden gesloten.

Een hoge mate van modulariteit bevordert de performanceoptimalisatiemogelijkheden. Binnen een modulair opgebouwde multi-tieromgeving kunnen applicatieservers worden geoptimaliseerd voor applicaties, databaseservers voor databases, enz.

Workload management
Binnen een two-tieromgeving is het moeilijk een effectieve en proactieve loadbalancing uit te voeren. De logica draait daar waar deze is ondergebracht en nergens anders.

Binnen een grootschalige omgeving moet de applicatieworkload flexibel kunnen worden verdeeld over meerdere processen en platformen. De verdeling moet kunnen plaatsvinden op basis van parameters en criteria, zoals de belasting per server, de verwerkingscapaciteit van de server en de prioriteit van de aanroepende en aangeroepen functionaliteit.

Fault tolerance
Binnen een two-tieromgeving stopt de verwerking indien een client of databaseserver onderuitgaat.

Binnen een grootschalige omgeving dienen sterkere waarborgen voor de continuïteit aanwezig te zijn. Het risico van het onderuit kunnen gaan moet worden onderkend en worden ondervangen. De infrastructuur moet kunnen onderkennen dat een applicatie- of databaseserver onbereikbaar is en moet aanroepen transparant verder leiden naar een nog beschikbare server.

Grote aantallen gebruikers
Two-tieroplossingen hebben moeite met grotere gebruikersgroepen.

Binnen grootschalige omgevingen moeten grote aantallen gebruikers kunnen worden verbonden met en bediend door applicatieservers en databases. Deze verbindingen moeten zo min mogelijk resources in beslag nemen.

Nadelen van TP-monitoren

TP-monitoren gaan een vaste plaats veroveren in het IT-landschap. Welke TP-monitoren uiteindelijk de concurrentiestrijd zullen winnen, is nog onduidelijk. Investeren in en ontwikkelen op basis van een TP-monitor maakt een bedrijf tot een early adaptor met alle risico's van dien.

De inzet van een TP-monitor gaat niet ten koste van de inzet van andere producten waardoor de aanschafprijs van een TP-monitor niet direct wordt gecompenseerd door een besparing. Daarbij zijn de prijzen van TP-monitoren afgestemd op de grootschaligheid van de omgeving waarbinnen ze worden ingezet. TP-monitoren brengen dus aanzienlijke investeringen met zich mee.

TP-monitoren zijn weliswaar 'proven technology' maar nog geen 'common technology'. Dit vertaalt zich in een schaarste aan ontwikkelaars met grondige kennis van zaken. Het vertaalt zich tevens in de beperkte ondersteuning van TP-monitoren door systeemontwerp- en -ontwikkeltools. Met een groot aantal populaire systeemontwerp- en -ontwikkeltools kan nog geen code worden geprogrammeerd of gegenereerd waarmee de mogelijkheden van TP-monitoren worden benut.

Hoewel voor de communicatie tussen de verschillende componenten binnen de bovenstaande architectuur een aantal open en gestandaardiseerde interfaces wordt gebruikt, zijn TP-monitoren (nog) niet gestandaardiseerd als product. De programmacode waarin TP-monitoraanroepen zijn opgenomen, zijn dan ook niet zonder meer overdraagbaar (portable) naar andere TP-monitoren. TP-monitoraanroepen zijn specifiek voor één bepaalde TP-monitor. Dus hoewel de transactielogica portable kan zijn, zal de applicatie dit niet zijn. Indien wordt besloten gebruik te maken van een TP-monitor, verbindt men zich dus aan een specifieke TP-monitor.

TP-monitoren, een uitwerking

Deze paragraaf gaat aan de hand van een voorbeeld dieper in op de werking van een TP-monitor. Daarnaast wordt de interne architectuur van een TP-monitor uitgediept waarbij de verschillende componenten waaruit een TP-monitor bestaat (en de communicatie hiertussen) de revue passeren.

De werking van een TP-monitor

TP-monitoren ondersteunen het three-tier- of multi-tier-model door te voorzien in een laagje 'Esperanto' in de vorm van een gemeenschappelijke API[6]. Deze standaardinterfaces maken het mogelijk dat de afzonderlijke client-, businesslogica- en databaseservices met elkaar kunnen communiceren via de TP-monitor.

Binnen een TP-monitoromgeving worden applicaties ontwikkeld als een verzameling van services die worden samengebonden in een weerspiegeling van de te ondersteunen bedrijfsprocessen. Een applicatie toont zich naar buiten toe als een samenhangend geheel van functionaliteit waarbinnen een gebruiker een taak uitvoert. Van binnen is de applicatie echter opgebouwd uit vele kleine stukjes applicatie (de services) die afzonderlijk kunnen worden aangeroepen. De services voeren zelfstandig of in combinatie met andere services taken uit voor een gebruiker. Zo zal bijvoorbeeld binnen een boekhoudkundig informatiesysteem altijd een debet- en creditfunctie worden aangeroepen binnen één logische boekingstransactie.

De communicatie tussen de verschillende onderdelen van de TP-monitor bestaat voornamelijk uit het doorgeven van kleine boodschappen. Bijvoorbeeld: een client stuurt een boodschap 'Betaal_Factuur(crediteur, Bedrag)' aan een server. Deze ontvangende (parent-)server start een transactie, en stuurt een boodschap aan child-server 1, 'Debiteer(Cred_Nr, Bedrag)' en child-server 2, 'Crediteer(Cred_Nr, Bedrag)'. De child-servers voeren de aangevraagde transacties uit en geven daarna de programmacontrole terug aan de aanroepende (parent-)server.

De TP-monitor behoudt het overzicht over de status van de verschillende bewerkingen die hebben plaatsgevonden. Pas wanneer de gehele transactie succesvol was, kan deze definitief worden gemaakt. Indien één van de bewerkingen niet succesvol was, dienen alle bewerkingen ongedaan te worden gemaakt. De laatste server uit het voorbeeld zal een commit[7]-opdracht opdragen aan de TP-monitor, die door de TP-monitor wordt doorgegeven aan alle betrokken resourcemanagers[8] (normaliter een databasemanagementsysteem). Wanneer alle resourcemanagers de commit kunnen uitvoeren, wordt de hele transactie gecommit of, indien dit niet mogelijk is, worden alle handelingen teruggedraaid tot de situatie voor de transactie werd opgestart. Hiermee is de integriteit van de transactie gewaarborgd.

De services zijn de codering van de businesslogica voor bedrijfsfuncties en zijn geschreven of gegenereerd in een ontwikkeltaal of -tool zoals Visual Basic, Cool:Gen, C of COBOL. De resourcemanager is meestal een databasemanagementsysteem zoals Oracle, Sybase of een ander soort dataopslag (bijvoorbeeld een plat ASCII-bestand of een queue).

De plaats van een TP-monitor

Een TP-monitor is een applicatie. De TP-monitorapplicatie draait binnen een netwerk en wacht op gebruikersaanvragen om deze af te kunnen laten handelen. Figuur 4 toont de TP-omgeving waarbinnen de TP-monitor, als een spin in zijn web, zijn werk uitvoert.

1 De clientapplicatie vraagt een transactie aan (bijvoorbeeld het bestellen van een boek). De clientapplicatie stuurt hiervoor een aanroepbericht naar een service.
2 De TP-monitor vangt alle clientaanvragen op en bepaalt waar de aangeroepen service draait en of de aanroep is geautoriseerd. Indien dezelfde service meerdere keren draait, bepaalt de TP-monitor aan welke kopie de aanvraag wordt doorgegeven.
3 De services die tezamen de functionaliteit van de applicatie vormen. De aangeroepen service gaat aan de slag en voert de inhoudelijke activiteiten uit die voor deze transactie zijn gecodeerd in de programmatuur. Wanneer binnen de programmacode is aangegeven dat bepaalde bewerkingen zijn uitbesteed, dan roept de service andere services aan om deelbewerkingen te verrichten.
4 Datasources bevatten gegevens en worden beheerd door resourcemanagers. Vaak zijn dit databasemanagementsystemen of queue-mechanismen. Gegevens kunnen afkomstig zijn van de clientapplicatie, van andere services of andere datasources. Gegevens kunnen ook naar al deze bronnen worden weggeschreven.

Andere functies van een TP-monitor

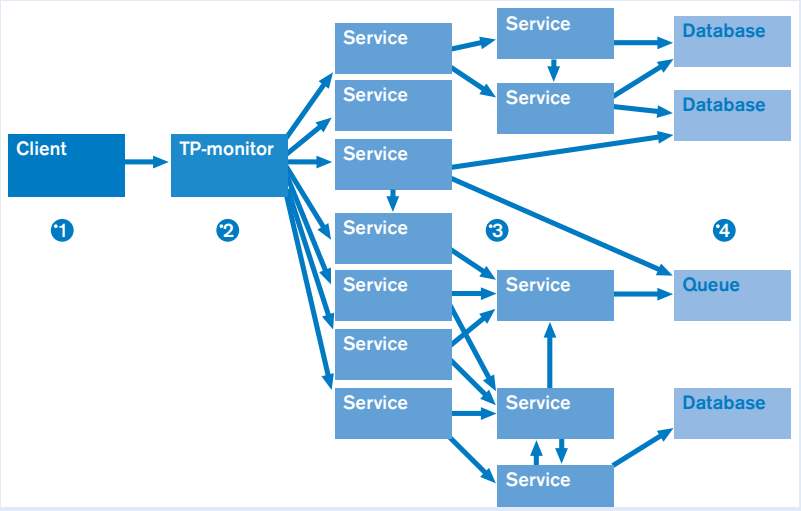
Een TP-monitor vervult nog meer functies, zoals blijkt uit de nu volgende beschrijvingen.

Security management

Een TP-monitor biedt ook security-managementfaciliteiten; de TP-monitor beheert de toegang tot de services en

6) API staat voor Application Programmable Interface; een verzameling van functieaanroepen wordt bedoeld waarmee een applicatie delen van haar functionaliteit beschikbaar stelt aan applicaties van derden. Dit in tegenstelling tot een user interface die functionaliteit beschikbaar stelt aan mensen.
7) Een commit is een commando waarmee een bewerking definitief wordt gemaakt.
8) Een resourcemanager is een component binnen de IT-infrastructuur die een bron (resource) beheert. Voorbeelden hiervan zijn een message queue en een databasemanagementsysteem.

Figuur 4. De plaats van de TP-monitor.



ondersteunt daarmee de vertrouwelijkheid binnen een informatiesysteem. Op welke wijze en met welke mate van striktheid dit gebeurt, is meestal instelbaar via parametrisering. Hiernavolgend worden de maximale beveiligingsmogelijkheden beschreven. Het gebruik van de maximale beveiligingsmogelijkheden brengt een additionele beheersinspanning met zich mee voor het nauw luisterende onderhoud van de autorisatiestructuren. In de praktijk wordt daarom niet altijd gebruikgemaakt van de maximale beveiligingsmogelijkheden.

Voordat een gebruikersaanvraag wordt afgehandeld, zal de (clientapplicatie van de) gebruiker toegang moeten krijgen tot de TP-monitor. De TP-monitor kan daarbij om identificatie en authenticatie vragen door middel van een gebruikersnaam en wachtwoord. Na het verkrijgen van toegang bepaalt de TP-monitor aan de hand van een autorisatietabel tot welke gebruikersgroepen de ingegeven gebruikersnaam behoort.

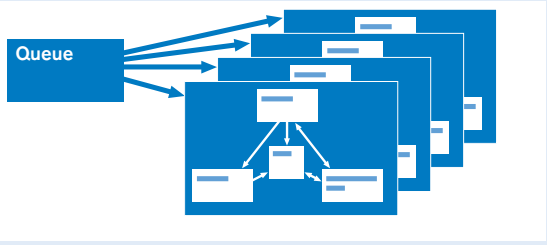
Binnen een TP-monitor kan per service worden aangegeven welke gebruiker(sgroep) toegang heeft tot de betreffende service. Iedere service bevat hiervoor een lijstje van gebruikers en gebruikersgroepen. Bij iedere aanroep van een service zal de TP-monitor controleren of de aanroepende gebruiker gerechtigd is de service aan te roepen.

Binnen de toegangsrechten op services wordt geen onderscheid gemaakt in lezen, schrijven of verwijderen. Een service is een applicatie en kan worden aangeroepen of niet worden aangeroepen. Het lezen, schrijven en verwijderen heeft betrekking op de functionaliteit van de service en voltrekt zich buiten het gezichtsveld van de TP-monitor.

Een TP-monitor ondersteunt controleerbaarheid door te loggen wie wanneer toegang heeft gezocht en met welk resultaat. Daar alle communicatie van een client met de achterliggende applicaties en data via de TP-monitor verloopt, hebben gebruikers geen rechtstreekse toegang nodig tot databases en applicatieservers.

System management

Een TP-monitor en de applicatieomgeving moeten even beheerbaar zijn als een database of een besturingssysteem (inclusief ondersteuning door tools). Deze functionaliteit weegt zwaar in grootschalige, complexe en multitieromgevingen, waarin vele en gerelateerde services kunnen draaien op meerdere fysieke locaties.



Figuur 5. Transactieaanvoer naar meerdere applicaties vanuit één queue.

TP-monitoren worden beheerd door middel van eigen system management tools of sluiten aan bij system management tools van derden. Logische en fysieke componenten binnen de IT-infrastructuur, zoals hardwareservers en services, worden grafisch weergegeven op een scherm en geven de system administrator een algeheel beeld van wat waar gebeurt. Naast configuratie- en statusgegevens worden ook performancegegevens en statistieken verzameld en gerapporteerd.

Fault-tolerancevoorzieningen

Fault tolerance wordt bereikt door transparantie van locaties en dynamische routeringsfaciliteiten. Hiermee wordt een transparante failover mogelijk in geval van een probleem met een server. Indien een aanvraag niet wordt gehonoreerd door een bepaalde server, worden nieuwe pogingen automatisch gerouteerd naar een andere server die kan antwoorden op de aanvraag. De failover en herroutering worden transparant uitgevoerd voor de client.

Meerdere kopieën van services

Indien een applicatie slechts op één plek draait, zullen alle clients gebruik moeten maken van dit applicatieproces. Hiermee ontstaan schaalbaarheidsproblemen en neemt de veerkracht (resilience) van het gehele systeem af. Eenzelfde applicatie kan meermalen en op verschillende plaatsen worden opgestart en worden voorzien van een 'portaal' waarin al het aangeboden werk wordt opgevangen en verdeeld. Het is daarmee mogelijk op enig moment meerdere kopieën van een applicatie (services) draaiende te hebben die worden gevoed vanuit één gemeenschappelijke queue met transacties (zie figuur 5).

Deze mogelijkheid wordt gebruikt ten behoeve van workload management en fault-tolerancevoorzieningen. Het daadwerkelijk aanwezige aantal kopieën van services kan dynamisch variëren tussen een geconfigureerd minimum en maximum. Aanvullende kopieën van de services worden gestart op basis van de queuelengte en worden gestopt wanneer ze langer dan een bepaalde tijd niet worden gebruikt. Door deze eigenschappen maakt een TP-monitor efficiënt gebruik van resources.

Teneinde de applicatiefunctionaliteit dynamisch te kunnen spreiden en managen op basis van de workload, kunnen wijzigingen dynamisch worden doorgevoerd zonder dat de TP-monitoromgeving hoeft te worden gestopt. Deze functionaliteit kan daarnaast worden toegepast om bijvoorbeeld transparante upgrades van servers uit te voeren door het opstarten van en routeren naar applicatieservices op andere servers terwijl een server of service wordt geüpgraded.

Workload management

De TP-monitor verzorgt het workload management van de aangeboden transacties over de beschikbare resources. Hiermee kan de tijdigheid van applicatiefuncties worden gestuurd. De TP-monitor besluit op basis van een aantal gegevens en een model naar welke server wordt gerouteerd wanneer meer dan één server aanwezig is. De beschikbare modellen kunnen variëren van een willekeurige keuze tot en met intelligente load balancing-

algoritmen waarbij wordt geprobeerd aanvragen toe te wijzen aan de server met de kleinste workload.

Bepaalde functies stellen hogere eisen aan de performance dan andere functies. Om prioriteiten te ondersteunen zijn op prioriteit gebaseerde routeringsschema's voorhanden. De routering kan tevens afhankelijk zijn van de inhoud van de data die moet worden gerouteerd. TP-monitoren ondersteunen deze functionaliteit onder de term Data Dependent Routing. Deze vorm van routering is een elementaire vorm van workflowmanagement.

Asynchrone queuing

Veel transacties zijn asynchroon van aard. Denk bijvoorbeeld aan het sturen van een ontvangstbevestiging van een e-mailbericht. Een gebruiker zal niet willen wachten op het moment van aankomst omdat het transport enige tijd vergt. De gebruiker zal tevreden zijn wanneer op een later tijdstip wordt gemeld dat het bericht is aangekomen.

Het vraag-en-antwoordmodel, waarbij een client na aanroep van een service blijft wachten op de afronding, is dus niet altijd de beste communicatiemethode. Niet vanuit het oogpunt van het te ondersteunen bedrijfsproces, maar ook niet vanuit het oogpunt van schaalbaarheid. Vraag-en-antwoordmechanismen kunnen namelijk onnodig lang beslag leggen op resources zoals memory en locks, hetgeen onwenselijk is vanuit het oogpunt van schaalbaarheid. Store-and-forwardmechanismen[9] zullen minder resources verbruiken en gedurende een kortere tijd.

TP-monitoren ondersteunen een asynchroon queuingmechanisme ten behoeve van asynchrone transacties. Dit wordt gerealiseerd door een queue in te zetten die wordt gemanaged als een resourcemanager. De queues worden bijgehouden in het geheugen of opgeslagen op een schijf wanneer ze beschermd tegen uitval moeten zijn.

Het managen van queues betekent overhead. De client hoeft echter niet langer te wachten op een transactie en ontvangt asynchroon de gewenste data (het antwoord) op een later moment. Dit model wordt momenteel nog slechts mondjesmaat toegepast maar heeft een groot potentieel. Behalve voor bedrijfsprocessen die inherent asynchroon zijn van karakter, kunnen queues ook worden gebruikt voor replicatie tussen databases, het vullen van datawarehouses en voor het aanmaken van audit logs.

TP-monitoren in de toekomst

Deze paragraaf werpt een blik in de toekomst van TP-monitoren.

Gedistribueerde systemen

De overgang van monolithische architecturen naar client-serverarchitecturen is een verandering van paradigma geweest waarbij applicaties werden opgesplitst in een client- en een servergedeelte. Momenteel treedt een nieuwe verandering van paradigma op, die van gedistribueerde systemen. Binnen gedistribueerde systemen worden de client- en de servergedeelten verder opgedeeld in componenten die samenwerken over netwerken heen.

In de markt zijn twee standaarden ontstaan voor gedistribueerde systemen: CORBA (Object Management groep) en DCOM (Microsoft). Beide bieden een infrastructuur waarbinnen objecten en componenten kunnen communiceren en samenwerken over verschillende typen netwerken en besturingssystemen heen. De CORBA-standaard wordt gestaag uitgebreid met specificaties, onder andere op het gebied van databasetechnologie en -services die reeds worden geboden door de huidige TP-monitoren.

De vraag is hoe client-serveromgevingen zullen worden beïnvloed door deze nieuwe ontwikkelingen. De CORBA-specificaties worden momenteel geïmplementeerd binnen Object Request Brokers (de middleware die alle communicatie en management binnen de gedistribueerde systemen regelt), soms door eigen ontwikkeling, soms door integratie met bijvoorbeeld bestaande databases en TP-monitoren zoals Tuxedo. Op deze wijze ontstaat een nieuw soort middleware dat proven technology aanbiedt in een nieuw concept. Object Resource Brokers worden daarmee een alternatief voor de traditionele TP-monitoren.

Ondersteuning van TP-monitoren

Van de zijde van databaseleveranciers, de leveranciers van kleinere TP-monitoren die de communicatie tussen clients en hun databases reguleren, worden de voordelen van een krachtige TP-monitor onderkend en worden koppelingen aangeboden naar commerciële TP-monitoren. Een leverancier als Oracle is bijvoorbeeld daarnaast bezig haar databasemanagementsysteem te voorzien van een CORBA-compliant interface waardoor haar databasemanagementsysteem kan integreren met TP-monitoren.

Van de zijde van systeemontwerp- en ontwikkelleveranciers is eenzelfde beweging in gang gezet. Leveranciers gaan over tot het ondersteunen van multi-tierarchitecturen en het ondersteunen van TP-monitoren als standaardcomponenten binnen een IT-infrastructuur. Een product als bijvoorbeeld Cool:Gen genereert vanuit ontwerpspecificaties programmatuur waarin de aanroepen naar een TP-monitor reeds zijn opgenomen.

De opkomst van TP-monitoren wordt tevens onderstreept door de groeiende bereidheid van leveranciers van systeemmanagementtools om het beheer van TP-monitoren mogelijk te maken vanuit deze tools.

9) Ook bekend onder de noemer 'fire & forget'.

Samenvatting en conclusie

Deze paragraaf geeft een samenvatting en conclusie over de mogelijkheden van multi-tierarchitecturen en de inzet van TP-monitoren bij het realiseren hiervan.

Schaalbaarheid, multi-tieromgevingen en de comeback van TP-monitoren

Monolithische en two-tierarchitecturen zijn beperkt schaalbaar. Steeds meer applicaties hebben belang bij een niveau van beschikbaarheid, performance, throughput, transparantie, beheersing en openheid waar traditionele architecturen niet in kunnen voorzien. Deze beperkingen van de monolithische en two-tierarchitectuur creëren een tendens naar oplossingen die zijn gebaseerd op een multi-tierarchitectuur.

Binnen de traditionele architecturen kan een aantal stappen worden genomen om tegemoet te komen aan de schaalbaarheidsbeperkingen. De architecturen zelf vormen echter een beperkende factor. Een structurele oplossing is nog niet voorhanden; hiervoor is de overgang naar een multi-tierarchitectuur nodig.

De technologie die deze multi-tierarchitectuur mogelijk maakt, wordt in het algemeen middleware genoemd en meer specifiek: een TP-monitor. Na het oplossen van gelijksoortige problemen in mainframeomgevingen, krijgen TP-monitoren nu de kans zich te bewijzen binnen midrangeomgevingen.

De inzet van een TP-monitor is geen trivialiteit; het is een overgang naar een nieuw systeemontwikkelingsparadigma.

TP-monitoren maken het mogelijk clientpresentatie te scheiden van de applicatielogica, en de datalogica en de verschillende componenten in te zetten op meerdere en verschillende soorten platformen. Transacties over meerdere en verschillende applicaties en datasources worden gecoördineerd door de TP-monitor. Daarnaast is geavanceerd workload management aanwezig voor het reguleren van de belasting en de performance en zijn workflowmanagementalgoritmen aanwezig om de flow van transacties te sturen.

De inzet van een TP-monitor is geen trivialiteit; het is een overgang naar een nieuw systeemontwikkelingsparadigma met alle nadelen van dien. De complexiteit van en de benodigde kennis omtrent de inzet van TP-monitoren houden een onmiddellijke toepassing nog tegen. In geval van nieuw- en herbouwtrajecten daarentegen is de inzet minder ingrijpend.

Een nadeel van de huidige TP-monitoren is dat het leverancierseigen producten zijn met leverancierseigen interfaces. Applicaties die gebruikmaken van een TP-monitor maken gebruik van de specifieke programmeercommando's van die TP-monitor, verliezen hun portabiliteit en binden daarmee een gebruiker aan een specifieke TP-monitor. In een markt die sterk in beweging is, tellen de risico's van een early adaptorship zwaar.

Daarnaast bestaan nog de voor de hand liggende afwegingen als de extra investering en de ondersteuning van een leverancier. Een TP-monitor is echter een bewezen technologie voor het realiseren van een grootschalige omgeving; de prijs die daaraan is verbonden, zal gezien de voordelen zeker overkomelijk blijken.

TP-monitoren zullen hun weg naar de markt vinden via bedrijven die behoefte hebben aan de mogelijkheden van deze nieuwe technieken en de bijbehorende overhead voor lief nemen. Bedrijven zullen TP-monitoren, in welke vorm dan ook, gaan invoeren als onderdeel van hun IT-infrastructuur en gebruik gaan maken van de toenemende functionaliteit en geavanceerdheid van deze producten. Deze omschakeling zal eerder een evolutionair dan een revolutionair karakter hebben.

Literatuur

[Bloo97]
Bloor Research Nederland, *Distributed Objects & Client/Server Computing*, 1997.

[EWP95]
Entersoft White Paper, *The Evolution to Middleware-Based, Multi-tier Distributed Computing*, 1995.

[EWP96a]
Entersoft White Paper, *TOPEND and Oracle*, 1996.

[EWP96b]
Entersoft White Paper, *TOPEND Product Overview*, 1996.

[Gray96]
J. Gray, A. Reuter, *Transaction Processing concepts and Techniques*, 1996.

[Hall96]
Carl L. Hall, *Building Client/Server Applications Using Tuxedo*, 1996.

[McFa99]
F.R. McFadden, J.A. Hollen, M.B. Prescott, *Modern Database Management*, 1999.

[Orfa94]
R. Orfali, D. Harkey, J. Edwards, *The essential Client/Server survival guide*, 1994.

[Orfa96]
R. Orfali, D. Harkey, J. Edwards, *The essential distributed objects survival guide*, 1996.

[OWP97]
Oracle White Paper, *Middleware directions, enabling the information age through network computing*, 1997.