Milleks on vaja tehingutöötluse monitori ?

On mitmeid võimalusi äritarkvara toetavaks töökindlaks vahevaraks (*middleware*):

- klient / server andmebaasid, millele asutakse lisama hajutatud rakenduste toetust (lisaks hajutatud andmete toetusele)
- tehingutöötluse monitorid (*transaction procerssing monitors*), mis toovad töökindla hajatöötlustehnoloogia suurarvutitelt (*mainframe*) klient / server platvormidele
- hulk uut vahevara, s.h. sõnumiedastus (*messaging*), grupivara (*groupware*), objektide nõudemaaklerid (*object request broker*) ja komponenttehnoloogiad

Kas need on lihtsalt erinevad viisid sama tehnoloogia pakkimiseks, või on olemas põhimõttelisi erinevusi, mis teevad ühe lähenemise paremaks teatud klassi kuuluvate rakenduste puhul?

Sissejuhatus

Moodsatele andmebaasiohjuritele lisatakse üha enam omadusi, mida võis varem leida vaid tehingutöötluse monitorides. Hiljutised uuendused on hajustehingud, salvestatud protseduurid ja partemad haldusvahendid. Samal ajal laieneb tehingutöötluse monitoride kasutamine klient / server süsteemides, lubades paremat töökindlust, skaleeritavust ja hallatavust.

Kas ma vajan tehingutöötluse monitori, et:

- installeerida klient / server rakendusi?
- pääseda juurde mitmele andmebaasiohjurile?
- kasutada järjekordi?

Kas tehingutöötlus monitor aitab:

- kui töökoormus tõuseb?
- kui kasutajate arv tõuseb?
- mingist teatud hetkest alates?

Milliseid teisi lahendusi on olemas?

Kasutajate nõuded

Vastamine nendele küsimustele pole kunagi lihtne olnud. Tehingutöötluse süsteemide arendajad seisavad täna paljus samade probleemide ees, mille ees on nad aastaid seisnud, lisandunud on ka uued nõudmised:

tehinguohjur, mis realiseerib ÜTEP (*ACID*) semantikat
skaleeritavad lahendused, et:
- teenendada suurt hulka paralleelseid kasutajaid
- tagada süsteemi ressursside effektiivset kasutamist
- võimaldada töökoormuse ja ressursside tasandamist

iseseisvalt väljatöötatud rakenduste ja uute andmetüüpide integreerimine
paremad rakenduste väljatöötamise ja süsteemide haldamise vahendid

Lahendused

Traditsioonilised lahendused nendele probleemidele sisaldasid nii andmebaasiohjureid kui ka tehingutöötluse monitore. Nendes süsteemides on mitmeid sarnaseid osi. Ehkki andmebaasiohjurid on põhiliselt andmeohjurid (*data manager*) ja tehingutöötluse monitorid protsessiohjurid (*process manager*), sisaldavad tegelikud tooted osi mõlemast. Nii on tehingutöötluse monitoride koosseisus anmeohjurid ja andmebaasiohjurite koosseisus protsessiohjurid. See viib segadustele ja mõlema rolli hägustumisele lahendatavas äriprobleemis.

Uued tehnoloogiad mis veelgi vett segavad:

- sõnumiedastussüsteemid (näit. IBM MQSeries), mis pakuvad ajast sõltumatut viisi rakenduste hajutamiseks (sageli nimetatakse neid ka vallas-tehingutöötluse (*off-line transaction processing*) süsteemideks)
- grupivara (näit. Lotus Notes), mis toetavad andmete jagamist suurele hulgale kasutajatele
- objektide nõudemaaklerid, milledel on lisatud tehinguraamistik (*transaction framework*) (näit. IBM SOM)
- komponendipõhised tehingutöötlussüsteemid (näit. IBM CBSeries ja MS OTS)

Tehinguohjur

Tehinguohjur on süsteemne tarkvara, mis lubab hulgal rakendustel jagada andmeid, säilitades viimaste terviklikuse. Algselt töötati tehinguohjurid välja juurdepääsuks tsentraliseeritud andmetele, aga nad on arenenud toetama ka juurdepääsu hajutatud andmetele. Tehiguohjur realiseerib *tehingu* kui hulga *taastatavaid* andmeid muutvate arvutuste mõistet, mis omavad ÜTEP omadust:

- *Ühiklikus* -- kõik muutused kas toimuvad või ei toimu ühikuna; andmed mida muudetakse tehingu käigus muutuvad tehingu piiril
- *Terviklikus* -- tehing säilitab andmete invariantsed omadused; tehing viib andmed ühest korrektsest seisundist teise korrektsesse seisundisse
- *Eraldatus* -- andmete vaheväärtused pole nähtavad teistele tehingutele; tehigud näivad toimuvat järjestiku isegi kui neid täidetakse paralleelselt
- *Püsivus* -- lõpetatud tehingu tulemus on püsiv; taastatavate andmete muudatused säilivad ka süsteemi või rakenduse vea korral

Tehingu mõistet toetavad nii andmebaasiohjurid kui ka tehingutöötluse monitorid.

Skaleeritavus

Skaleeritavus on tehingutöötluse süsteemide omadus, mis lubab kasvu mitmes mõõtmes:

- süsteemi kasutajate arv
- ajaühikus sooritatud tehingute arv
- töödeldud andmete maht

Skaleeruv süsteem lubab süsteemi võimsust nendes mõõtmetes kasvatada muutmata rakendusi.

Jätkuv kättesaadavus

Tehingutöötluse süsteemid toetavad ärikriitilisi rakendusi, seega peavad nad olema võimelised tegutsema ükskõik kunas ja kus äritegevus toimub. See vajab:

- planeeritud seisakute vähesust või puudumist
- planeerimata seisakute vähesust
- kiiret taastumist väljalangemistest, mis siiski toimuvad

Tööriistad

Kuna tehnoloogia hind langeb, on omamiskulutused määratud kulutustega inimestele. Need sisaldavad nii väljatöötamist kui ka haldamist:

- kiire rakenduste väljatöötamine on vältimatu saavutamaks ja säilitamaks edumaad ärilises tegevuses (võitlusvõimet); siia kuulub lõpkasutaja tööviljakus, ühildumine / ühendumine teiste tööriistadega ja effektiivsete rakenduste lihtne väljatöötamine
- installeeritud süsteemi haldamine ei tohi nõuda suurt hulka inimesi; rakenduste installerimine, konfigureerimine ja kasutamine peab olema lihtne ning nõudma minimaalselt oskustööd

Andmebaasiohjurid

Järgnevalt vaatleme lähemalt kuidas andmebaasiohjurid neid nõudmisi täidavad.

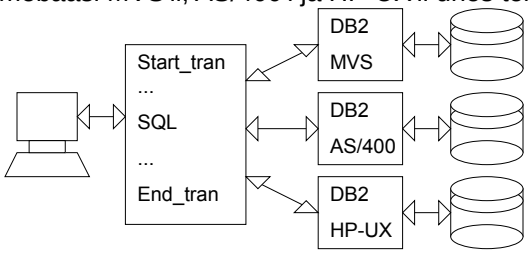
Tehinguohje

Andmebaasiohjurid on alati tehinguid realiseerinud, aga need on olnud seotud antud andmebaasisüsteemi ressurssidega. Kui tehingus peab osalema rohkem kui üks andmebaas tekib vajadus välise tehinguohjuri (tavaliselt tehingutöötluse monitori koosseisus) järele. Tänapäeva andmebaasiohjurid on seda kitsendust lõdvendanud vähendades vajadust välise tehinguohjuri järele. Tänapäeval toetavad andmebaasiohjurid läbipaistvat juurdepääsu andmetele, mida hoitakse paljudel erinevatel platvormidel.

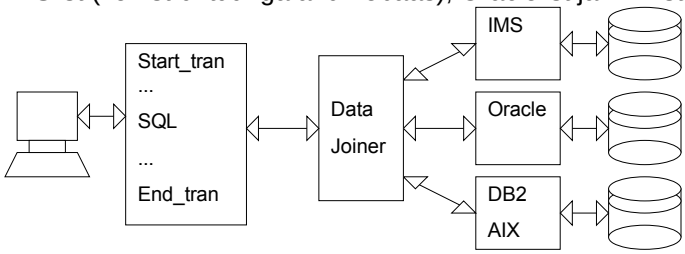
Läbi erinevate hajutamisprotokollide võib andmebaasi klient pääseda juurde andmetele erinevates relatsioonilistes andmebaasides (näit. DB2 Windows'i klient võib läbipaistvalt ühenduda DB2 külge, mis jookseb SP2 klastril, MVS serveril või Sun'il Solaris'e all ja sooritada tehinguid neist igaühega). Juurdepääs kõikidele andmebaasidele toimub nime järgi, teadmata tegeliku serveri asukohta või sideprotokolli, mille abil tuleb ühenduda.

Hajustehingud

Tehingud, mis hõlmavad mitmeid süsteeme on samuti toetatud. Sama andmebaasi klient võib uuendada DB2 andmebaasi MVS'il, AS/400'l ja HP-UX'il ühes tehingus.



Andmed mitte-relatsioonilistest andmebaasidest ja ühendused (*join*) heterogeensetest allikatest on samuti võimalikud. IBM'i *DataJoiner* teeb võimalikuks saada andmeid mitmetest allikatest, kombineerida neid ja esitada kasutajale nii nagu nad pärineksid ühest andmebaasist. Näiteks võib ühendada ja uuendada andmeid IMS'ist (võrkstruktuuriga andmebaas), Oracle'ist ja DB2'st AIX'il.



Tulevikusuunad

Andmebaasiohjurite funktsionaalsus laieneb pidevalt. Andmebaasiohjurid realiseerivad X/Open TX protokolli tehingute alustamiseks ja lõpetamiseks, mis lubab neil koordineerida tehinguid mistahes XA'le vastava ressursihalduriga (näit. DB2, Informix, Oracle, mitte-relatsioonilised andmebaasid ja järjekorrasüsteemid).

Skaleeritavus

Et parandada skaleeritavust keskenduvad tänapäevased andmebaasiohjurid kolmele valdkonnale:

- salvestatud protseduurid -- rakenduse koodi täitmine serveris vähendab side koormust, parandades vastuseaega ja läbilaskevõimet
- paljulõngalisus (*multi-threading*) -- kliendi effektiivsem esitus serveris lubab kasutada paljuprotsessorilisust ja toetada suuremat hulka kliente
- koormuse tasandamine -- kliendi päringute inteligentne hajutamine sarnaste serverite hulgal lubab kasvatada süsteemi lisades servereid

Salvestatud protseduurid

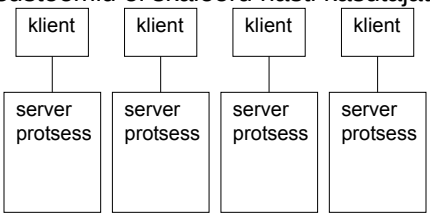
Salvestatud protseduurid pakuvad võimalust täita rakenduse koodi serveril vastavalt kliendilt tulnud päringule ja effektiivselt täita tehing ilma edasise andmevahetuseta võrgus (sarnased protseduride kaugkutsele (*RPC*)). Mõned andmebaasiohjurid lubavad salvestatud protseduuride realiseerimist mistahes programmeerimiskeeles, teised lubavad kasutada vaid antud tootesse kuuluvat 4 põlvkonna keelt.

Salvestatud protseduurid vähendavad võrguliiklust, soosivad andmete kapseldamist ja rakenduste sõltumatust andmetest, salvestades nii andmed kui ka neid töötlevad protseduurid andmebaasiserveris.

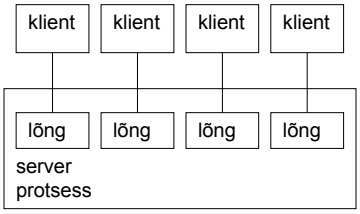
Järgmine samm sellel teel on n.n. universaalsete andmebaasiohjurite tekkimine, mis lubavad laiendada andmebaasi salvestatavate andmetüüpide valikut uute andmetüüpide realiseerimisega (DB2 -- *extenders*, Informix -- *datablades*). Selline uus andmetüüp on tunduvalt paremini integreeritud andmebaasiohjurisse, võimaldades kasutada teda nii andmete kirjelduskeeltes (*DDL*) kui ka päringukeeltes (*SQL*). Samuti seob ta tugevamalt andmed neid töötlevate protseduuridega ja võimaldab paremat korduvkasutamist, kehtestades üldise standardi andmeid töötlevatele protseduuridele.

Paljulõngalisus

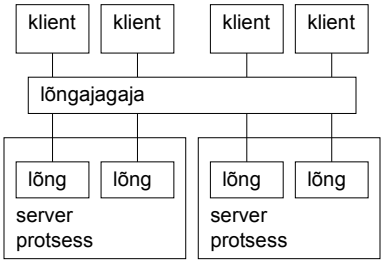
Varajased andmebaasiohjurid vajasid iga ühendunud kasutaja kohta ühte operatsioonisüsteemi protsessi. Sellised süsteemid ei skaleeru hästi kasutajate lisamisel, kuna protsessid on tavaliselt ressursimahukad.



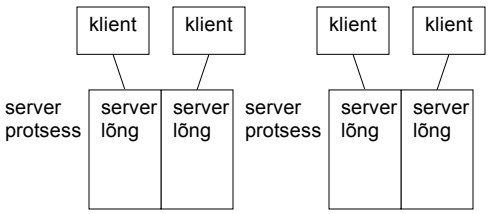
Järgnevalt pakuti ühe protsessi sees rakenduse tasemel realiseeritud lõngajagajat (*thread scheduler*). Sellise lahenduse korral tegeleb iga kasutajaga ühe protsessi sees olev lõng. Sellised süsteemid vajasid oma eelkäijatest oluliselt vähem süsteemi ressursse, skaleerudes küllalt hästi kasutajate hulga lisamisel. Kuna aga sellise lahenduse korral kõik lõngad jooksevad ühes operatsioonisüsteemi protsessis ei saa need süsteemid tulu sümeetrilistest multiprotsessorsüsteemidest (SMP).



Nüüd tekkisid andmebaasiohjurid, mis olid võimelised kujutama rakenduse tasemel realiseeritud lõngad mitte ühele vaid tervele hulgale operatsioonisüsteemi protsessidele, võimaldades kasutada SMP süsteeme.



Kuna tänapäeval on lõngad tavaliselt realiseeritud ka operatsioonisüsteemides, kasutavad kaasaegsed andmebaasiohjurid tavaliselt operatsioonisüsteemi lõngu rakenduse tasemel realiseeritud lõngade asemel, vältides ebaefektiivsust mis tuleneb kahe erineva lõngajagaja koordineerimatusest.



Kuna kõik operatsioonisüsteemid ei sisalda ikka veel lõngade toetust, piirab viimane lahendus andmebaasiohjuri võimaliku platvormide hulka.

Koormuse tasandamine

Olgugi, et andmebaasiohjurite juures on tehtud suuri edusamme, on koormuse tasandamine tehingutöötluse monitoride, spetsiaalrakenduste ja järjekorrasüsteemide põline ala. Tänapäeva andembaasiohjurid toetavad kahjuks nõrgalt koormuse jagamist mitme serveri vahel.

Suurem osa andmebaasisüsteeme sisaldavad ennustussüsteeme ja server koormusepiirajaid, mis jääb alla tehingutöötluse monitoride poolt pakutavale funktsionaalsusele.

Tööriistad

Andmebaasiohjurid on hästi kaetud mitmesuguste rakenduste tegemise tööriistade toega. Tänapäeval on raskuspunkt üle kandunud süsteemide haldamisele. Suurt edu on saavutatud effektiivsuse monitoride ja administreerimistööriistade juures. On olemas vahendid mitmesuguse statistika kogumiseks, analüüsimiseks ja graafilised vahendid tulemuste esitamiseks.

Andmebaaside haldamisvahendid on teine kiirelt arenev ala. Kiired ladurid võimaldavad andmete laadimist baasi kiirustega kuni 2Gb/tunnis ühe klastrisõlme kohta.

Hoolimata kõigest sellest on hulga andmebaaside haldamine siiski keerukas. IBM pakub selles vallas toodet *DataHub*, mis võimaldab hallata tsentraalselt heterogeenseid andmebaase (DB2, Oracle, Sybase).

Transaktsioonitöötluse monitorid

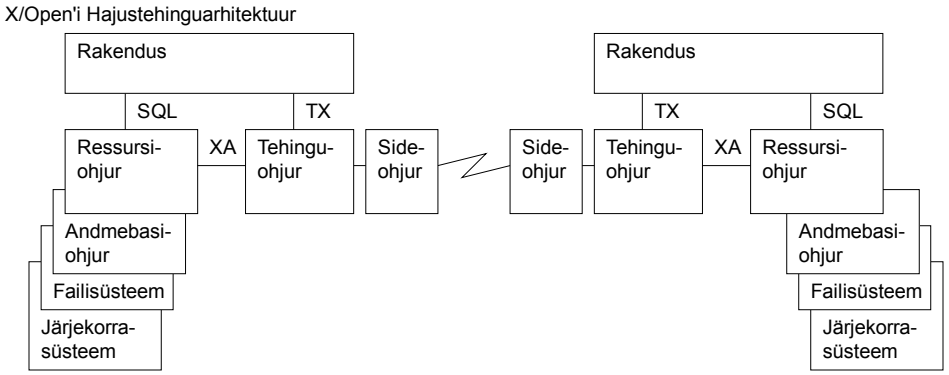
Tehingutöötluse monitor pakub töökeskonna serveril jooksvale korduvalt kasutatavale rakendusele. Sellised rakendused toetavad äritegevust suures osas maailmas. Sellised süsteemid on sageli kriitilise tähtsusega, peavad töötlema suuri andmemahtusid, teenendama suurt hulka kasutajaid ja nende vead võivad olla olulise mõjuga äritegevusele ja selle jätkumisele.

Tehinguohjamine

Alates oma tekkimisest on tehingutöötluse monitorid pakkunud tehinguid, mis uuendavad jagatud ressursse ja nagu ka andmebaasiohjurid, on nad läbi käinud sarnase arengu. Alguses töötas tehingutöötluse monitor vaid koos teatud, temaga ühtesobiva failisüsteemiga või andmebaasiohjuriga. Tehigute terviklikus aga sõltus andmebaasiohjuri ÜTEP omadustest.

Mitmete sidussüstseemide (*on-line systems*) installeerimisel tekkis tavaliselt vajadus kasutada mitmeid andmebaasiohjureid. See viis omakorda rakendusteni, mis kasutasid andmeid mitmest andmebaasiohjurist ja seega polnud enam võimalik tagada andmete terviklikust kasutades vaid andmebaasiohjurit, kuna ÜTEP omadused olid vajalikud tervel tehingul.

Väline tehinguohjur, mis sisaldub tehingutöötluse monitoris, tagab et kõikide andmebaaside tehingud toimuvad või jäävad toimumata. Kahefaasiline kinnitus (*two phase commit*) protokoll määrab liidese tehinguohjuri mis koordinerib ja ressursiohjurite (*resource manager*) mis osalevad ühises tehingus vahel. Selline on näiteks X/Open'i hajustehingu arhitektuuris XA liides.



Olgugi, et tänapäeval suudavad andmebaasiohjurid juba koordinerida tehinguid, mis haaravad mitut andmebaasi, puudub neil ikka veel toetus selliste tehingute koordineerimiseks, mis haaravad mitmeid heterogeenseid (eritüübilisi) andmebaase (andmebaasiohjureid).

Hajustehingud

Hajussüsteemide kasutamise laienemine tekitas vajaduse uuendada andmeid erinevates arvutivõrgu sõlmedes. See on lisanud uue mõõtme andmete terviklikuse probleemile -- tehinguohjur ei pruugi teada, kas kaugressursiohjur on lõpetanud tellitud tegevuse. Selle probleemi lahendamiseks tekkisid uued protokollid, mis kasutavad sideohjurit (*communication manager*) kui kaugressursiohjuri asemiku (*proxy*). Põhilisteks arhitektuurideks siin on:

- SNA APPC
- ISO OSI/TP standard
- tehingu kaugkutse TRPC

Vähemalt üks ülaltoodud arhitektuuridest on kasutusel suuremas osas tehingutöötluse monitorides, võimaldades mitmeid arvutivõrgu sõlmi haaravate hajusrakenduste tööd ja tagades andmete terviklikuse. Suurem osa andmebaasiohjuried toetab samuti hajustehinguid, kuid teeb seda oma sisemisi protokolle kasutades, piirates rakenduste hajutamist homogeensete keskkondadega.

Tulevikusuunad

Kui äriprotsesse kirjeldada eelnevalt defineeritud alam-protsesside abil, viib see välja sisalduvate tehinguteni (*nested transactions*). Sisalduvad tehingud defineerivad tehingute hierarhia, milles kõige ülemine tehing võb õnnestuda ühe või mitme järgmise astme tehingu õnnestumisel. See on eriti vajalik objektsüsteemide puhul, kus alamtehingud saab kapseldada taastatavatesse objektidesse (*recoverable objects*).

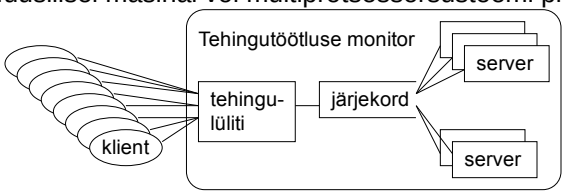
Sõnumiedastussüsteemid kasutavad kompositsionni, et toetada pikki tehinguid. Tänapäeval realiseeritakse see otse rakenduse loogikas aga tuleviku süsteemid lubavad alamtehingute tühistamist isegi peale nende edukat lõppu.

Skaleeritavus

Algusest peale on tehingutöötluse monitorid pakkunud vähe ressursinõudlikku lõngasüsteemi, mis lubas kasutajaga ühendatud rakenduse töid effektiivselt jaotada. See on olnud vastupidine paketttöötluse- või ajajaotussüsteemidele, mis tavaliselt kasutavad vähemalt ühte operatsioonisüsteemi protsessi ühendunud kasutaja kohta. Selle tulemusena võimaldavad tehingutöötluse monitorid vähemalt suurusjärgu võrra rohkem ühendunud kasutajaid.

Varajased tehingutöötluse monitorid realiseerisid lõngad rakenduse tasemel ühe operatsioonisüsteemi protsessi sees ja ei suutnud seega ära kasutada multiprotsessorsüsteeme. Edasine areng tõi kaasa operatsioonisüsteemi lõngade kasutamise mitmel erineval viisil. Juuresolev pilt näitab kuidas tehingute marsruuter mis sisaldub ühes protsessis suunab tehingud täitmiseks rakendusserverite puuli, mis on eellaetud, et saavutada kasutaja maksimaalselt kiiret teenindamist. Iga rakendusserver võib töötada

erineval füüsilisel masinal või multiprotsessorsüsteemi protsessoril.



Koormuse tasandamine

Varajased ruutimisalgoritmid olid staatilised ja küllaltki lihtsad. Konkreetne tehing konkreetselt kasutajalt marsruuditi alati samale rakendusserverile hoolimata serveri olekust. Kui servervälja langes, hakkasid selle serveri tehingud kogunema ja jäid tegemata.

Tänapäeval on tehingutöötluse monitorides intelligentsed marsruuterid, mis dünaamiliselt marsruudivad tehingud, vastavalt serverite jälgimisel kogunenud statistikast, "parimale" rakendusserverile.

IBM S/390 Sysplex klastril töötav CICS/ESA tehingutöötluse monitor näiteks parandab pidevalt oma marsruutimisalgoritmi vastavalt kogutud statistikale ja lubab administraatoril hallata süsteemi läbilaskevõimet ja vastuseaega vastavalt erinevatele poliitikatele. Ülekoormatud või väljalangenud rakendusserveritele määratud tehingud marsruuditakse automaatselt ümber korras serveritele, võimaldades süsteemil vigadest toibuda.

Tööriistad

Suurema osa oma ajaloo vältel on tehingutöötluse monitore kasutatud koos 3 põlvkonna keeltega nagu C, COBOL ja PL/I aga samuti ka koos 4 põlvkonna keeltega. Mitmed neist keeltest on nüüd kättesaadavad ka personaalarvutitel ning Unix'i platvormidel. Sageli on nad täiendatud mitmekülgsete programmeerimiskeskkondadega ja on effektiivsed vahendid tehingitöötluse rakenduste loomiseks. Mitmeid populaarseid visuaalse programmeerimise vahendeid on võimalik kasutada tehingutöötluse monitoridega samuti nagu andmebaasiohjuritega. Samuti on üha suurenev hulk OO keeli nagu C++ ja Smalltalk varustatud vahenditega rakenduste tegemiseks tehingutöötluse monitoridele.

Uued tehingutöötluse süsteemid on sageli realiseeritud klastritena, kasutades kohtvõrkude ja multiprotsessorsüsteemide tehnoloogiaid. See aga toob kaasa süsteemide haldamise keerukuse tõusu. Ressursside defineerimine, effektiivsuse monitooring ja juhttegevused peavad toimuma klastri igas sõlmes. Uutes tehingutöötluse monitorides on hakanud ilmuma tugi mis lubab sellised tegevused teostada ühel klastri sõlmel ja automaatselt paljundada seda tegevust üle kogu klastri. Vajalikuks osutub:

- süsteemi defineerimine ühest punktist
- süsteemi jälgimine ühest punktist
- süsteemi juhtimine ühest punktist

Tehnoloogiate kombineerimine

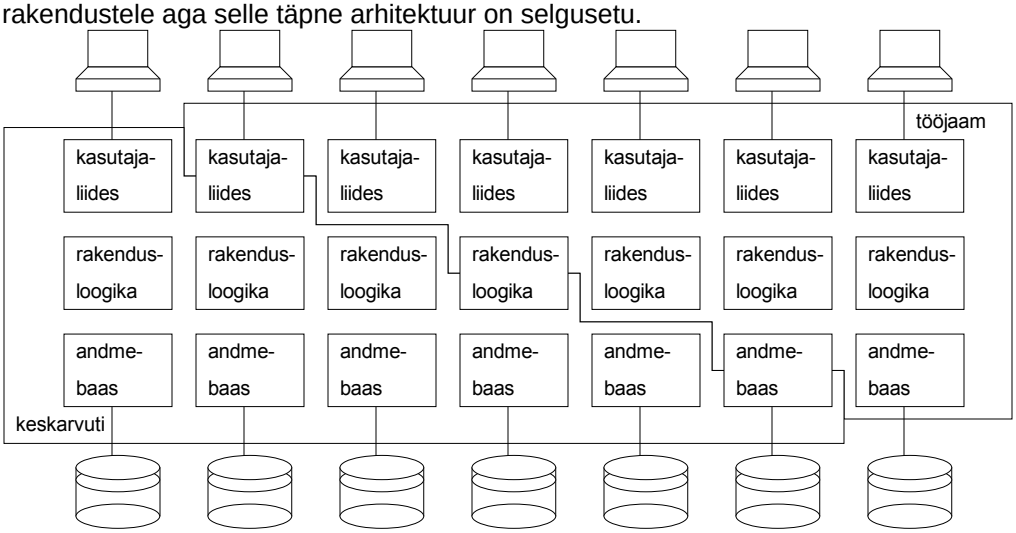
Selgelt on näha alad, kus tehingutöötluse monitoride ja andmebaasiohjurite kooskasutamine annab tuntavat võitu.

Kolmekihilised rakendused

Kolmekihiliste rakenduste (*three-tier applications*) tekkimine tagab, et sõltumatud tehgingutöötluse monitorid jäävad tehingutöötluse süsteemides tähtsale kohale. Mitmed rakendused on tänapäeval realiseeritud kolme kihilistena:

kasutajaliides (sageli graafiline)
rakendusloogika (ärifunktsionaalsus)
andmebaas

Tarkvaratööstuses on üldiseks arusaamaks, et kolmekihiline mudel on optimaalne tehingutöötluse rakendustele aga selle täpne arhitektuur on selgusetu.



Mitmed tarkvaratootjad on isegi lahenduse käigus välja töötanud oma enda keskvara, tuntuim näide on neist SAP/R3. SAP'i disainerid on oma süsteemi keskmise taseme (rakendusloogika) jaoks kirjutanud lihtsa tehingutöötluse monitori. Selline ise realiseeritud tehingutöötluse monitor aga ei suuda funktsionaalsuselt võistelda spetsiaalse keskvaraga.

On oodata, et ka teised rakenduste tootjad vajavad üha enam selliseid tehingutöötluse monitori teenuseid, nagu skaleeruvus ja koormuse tasandamine, ning kasutavad välist toodet nende teenuste saamiseks, nii nagu nad kasutavad andmebaasiohjureid.

Turvalisus

Turvalisus on tähtsaks teemaks tehingutöötluse süsteemides aga sageli ei seostata seda skaleeruvusega. Suure hulga kasutajate administreerimine tekitab teatud probleeme nii andmebaasiohjurites kui ka kui ka tehingutöötluse monitorides. Traditsiooniliselt toimub kasutaja autentimine igas serveris kasutajatunnuse ja parooli alusel. See muutub kiirelt haldamatuks, kui tuhanded kasutajad peavad kasutama sadu rakendusservereid, mis omakorda kasutavad kümneid andmebaasiservereid.

Effektiivne turvasüsteem nõuab, et nii rakendusserver kui ka andmebaasiohjur omaksid ühist turvastrateegiat. Parim kandidaat selleks tundub olevat DCE, kus iga kasutaja autenditakse üks kord oma lokaalse serveri poolt ja edasine autentimine teistes serverites toimub krüptitud "piletite" alusel. See tekitab illusiooni ühest ühisest võrku sissenemisest ja vähendab oluliselt administreerimisega seotud töömahtu.

Paralleeltöötlus

Multiprotsessorsüsteemid ja paralleelsüsteemid on hiljuti võetud kasutusele otsustuste toetusel, pakkudes olulist effektiivsuse tõusu SQL päringute paralleelse täitmisega andmebaasides, mis on jagatud klastri sõlmede vahel.

Kasutades multiprotsessor- või paralleelsüsteeme võivad tehingutöötluse rakendused töötada väga suurte koormustega ja olla peaaegu lõputult skaleeruvad. Paralleelsüsteemide kasutamise võtmeks on koormuse tasandamise algoritmide uus põlvkond, mis töötavad nii kliendil kui ka serveril. Klient marsruudib tehingud dünaamiliselt "parimale" serverile klastris, s.t. sellisele, mis hoiab tehingus vajaminevaid andmeid. Sellisel juhul muutuvad pöördused andmete poole lokaalseteks, suurendades kogu süsteemi kiirust ja läbilaskevõimet. Sellised süsteemid skaleeruvad lineaarselt sadade serveriteni.

Kokkuvõte

Oleme vaadelnud, kuidas andmebaasiohjurid ja tehingutöötluse monitorid lahendavad tehingutöötlusele esitatavaid nõudmisi ja kuidas nende funktsioonid selles osas ühte sulavad. Kas tehingutöötluse monitorid kaovad ja nende funktsioonid võetakse üle järgmise põlvkonna andmebaasiohjurite poolt?

- Välise tehinguohje puhul on see väga tõenäoline; on oodata, et tehinguohjur on juba järgneva põlvkonna operatsioonisüsteemide koostisosa, olles kasutatav kõikide rakenduste juures mitte ainult tehingutöötluses.
- Skaleeruvuse suhtes mitte kunagi (vähemalt mitte suurte ja väga suurte süsteemide juures); tehingutöötluse monitoridel on edaspidigi parim skaleeruvus, eriti paralleelsetel süsteemidel nagu IBM'i S/390 Sysplex ja SP2 ning DEC'i Alpha klaster; väikeste süsteemide puhul piisab tavaliselt andmebaasiohjurisse ehitatud tehingutöötluse vahenditest.
- Tööriistatde osas ilmselt mitte; süsteemide haldamine on jätkuvalt kriitiline koht suurte hajussüsteemide juures ja selles on tehingutöötluse monitorid parimad. Rakenduste väljatöötamise vahendite osas on andmebaasiohjurid ja tehingutöötluse monitorid võrdsed.

Uus, veel avamata potentsiaaliga vahevara on objektide nõudemaakler, mis samuti andmebaasiohjuritele on hakanud saama tehingutöötluseks vajaliku funktsionaalsust, ning toetab tehinguid komponentidega, rõhutades ärireeglite ja äriprotsesside kapseldamist komponentideks.

- Kas objektide nõudemaaklerid suudavad laienede funktsionaalselt, et edukalt täita ärikriitilistele tehingutöötlussüsteemidele esitatavaid nõudeid või laieneb tehingutöötluse monitoride toetus tarkvara komponentidele?
- Samuti, kas moodsad andmebaasiohjurid, püüdes vältida protsesside andmetest eraldamisel tekkinud probleeme, arenevad objektide nõudemaaklerite suunas?

On enam kui tõenäoline, et need kolm mingil viisil kokku sulavad, aga kas sellest saab alguse uus n.n. featürismi laine, kus tekivad massivsed suletud süsteemid, mis püüavad võimalikult palju funktsioone endasse haarata või suudetakse leppida kokku X/Open'i XA sarnased standardid, mis võimaldavad kasutajatel süsteeme komponeerida vajaliku hinna ja funktsionaalsuse suhtega komponentidest sellele saab vastata alles tagantjärele.