Optimalisering van file allocatie in gedistribueerde database systemen

Optimalisering van file allocatie in
gedistribueerde database systemen

by

A.J. Seebregts

87/19

december 1987

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science of Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author or the editor.

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB Eindhoven
The Netherlands

editor: F.A.J. van Neerven

OPTIMALISERING VAN FILE ALLOCATIE IN GEDISTRIBUEERDE DATABASE SYSTEMEN :

literatuuroverzicht.

auteur: A.J.Seebregts

Technische Universiteit Eindhoven, oktober 1987

Voorwoord.

Dit rapport is het resultaat van een stage onder begeleiding van prof.dr. K.M. van Hee en dr. A.T.M. Aerts. De opdracht was een literatuuronderzoek te doen op het gebied van file allocatie in gedistribueerde database systemen met betrekking tot de optimalisering van kosten. Daarnaast werd een eigen probleem en model op dit gebied geformuleerd met als doel een aantal verschillende aspecten op een meer overkoepelende wijze in zich te verenigen.

Het resultaat van het literatuuronderzoek is te vinden in de hoofdstukken 1 tot en met 3, het eigen model in hoofdstuk 4 en algemene conclusies in hoofdstuk 5.

INHOUDSOPGAVE.

1. INLEIDING.

Een gedistribueerd **database systeem** kan worden voorgesteld als een netwerk van sites en kanalen. In de sites bevinden zich computers en kunnen **files** worden opgeslagen. Een file kan eventueel gepartitioneerd worden (levert **fragmenten** van een file op) en op meer dan één site worden opgeslagen (levert **copieën** op). Gebruikers geven vanuit de sites opdracht tot transacties op de files. Deze transacties kunnen queries of updates zijn, of een combinatie daarvan. Zo'n query en update hebben een oorsprong, dat is de site waarvandaan de **query** en **update** wordt gegenereerd. Dit wordt ook wel de **host-site** genoemd. De benodigde datatransporten worden verzorgd door de kanalen. Aan deze transporten en aan het manipuleren en opslaan van files kunnen kosten worden verbonden. Een site kan een bepaalde opslagcapaciteit hebben. Verwerkingscapaciteiten, dat zijn de maximale hoeveelheden eenheden van gegevens die per tijdseenheid kunnen worden verwerkt, kunnen zijn toegekend aan zowel sites als kanalen.
Het probleem is nu, gegeven de frequentie, oorsprong en omvang van de benodigde datatransporten voor de transacties, hoe de files aan de sites toe te wijzen teneinde de genoemde kosten te minimaliseren. Daarbij kunnen naast de capaciteiten ook performance criteria als responstijden op queries en filebeschikbaarheid als restricties voorkomen.

Vooraf aan het literatuuroverzicht zullen belangrijke facetten van dit file allocatie probleem (afk. FAP) worden toegelicht.

Objecten.

De te alloceren objecten zijn **files** (of fragmenten daarvan). Soms wordt onderscheid gemaakt tussen **programma- en datafiles**: om toegang tot een datafile te krijgen, moet eerst een programma worden opgestart dat mogelijk niet in de host-site of in de site van de betreffende datafile aanwezig is. Per object zijn in principe **copieën** mogelijk.

Transacties.

Mogelijke transacties zijn queries en updates of een combinatie daarvan. Een query kan slechts één file raadplegen of verschillende files (resp. een **single-file en multi-file query** genoemd). Bij zo'n single-file query vindt er een transport van gegevens plaats van de site waar de file zich bevindt naar de host-site. Initiatieboodschappen van host-site naar de file-site worden veelal verwaarloosd; is dit niet zo, dan spreken we van **heen- en terugverkeer**. Bij een multi-file query vinden mogelijk ook datatransporten plaats tussen de verschillende files voordat het eindresultaat in de host-site beschikbaar is (denk

bijv. aan joins). Bij zo'n multi-file query zal een bepaalde **verwerkingswijze** horen, dat is een **schedule** van deeloperaties en een routering van datapakketten. Dit kan worden bepaald met een zgn. **query processing algorithme** (qpa). De optimale verwerkingswijze is afhankelijk van de file allocatie, waarvan de optimalisatie weer afhangt van de verwerkingswijze. Dit is een circulair probleem (en complex): veelal worden de beide problemen gescheiden opgelost, wat leidt tot een evt. suboptimale totale oplossing.Veelal wordt om een 'optimale' file allocatie te bepalen een **eenvoudige verwerkingswijze** genomen, dat wil zeggen dat de multi-file query als een verzameling single-file queries beschouwd wordt en als dusdanig zal worden verwerkt.
Een **update** heeft betrekking op een file. Er vindt een datatransport plaats van de host-site naar de file-site. Bij de aanwezigheid van copieën zullen deze allemaal worden 'geupdated'. Veelal neemt men aan dat er voor elke copie een transport plaatsvindt door het kanaal van de host-site naar de file-site; het totale transport voor deze copieën kan echter eventueel worden gereduceerd door 'gemeenschappelijke' kanalen te benutten. Hier geldt dus een analoge situatie als voor queries met betrekking tot de verwerkingswijze. Ook wordt terugverkeer (b.v. een bevestigingsboodschap) veelal verwaarloosd.

Kosten.

Deze kunnen bestaan uit **transportkosten**, afhankelijk van de hoeveelheid gegevens en/of van het kanaal; uit **opslagkosten**, afhankelijk van (de omvang van) de file en/of van de site; en uit **siteverwerkingskosten** om een file in een site te raadplegen (b.v. **access costs**).

Capaciteiten.

Capaciteiten zijn een **opslagcapaciteit**, afhankelijk van de site; een **kanaalcapaciteit** en **siteverwerkingscapaciteit**, dat zijn respectievelijk de maximale hoeveelheden transporteenheden van gegevens en maximale eenheden verwerkingseenheden die per tijdseenheid kunnen worden verwerkt.

Performance criteria.

Naast de systeemkosten kunnen de systeemperformance grootheden van belang zijn. Dit zijn onder andere: **responstijden** (op queries), en de **filebeschikbaarheid**, dat is de kans dat een te raadplegen file beschikbaar is, en hangt af van de topologie van het netwerk, van de kansverdelingen waarmee kanalen en sites tijdelijk niet beschikbaar zijn, en van de plaats van de copieën.

File onafhankelijkheid/Complexiteit.

Het alloceren van copieën van één file over n sites is een probleem van de orde $O(2^n)$; voor m files ten hoogste van orde $O(2^{nm})$ als men deze niet onafhankelijk (d.w.z. dat de allocatie van de ene file wordt beinvloed of beperkt door een andere) kan alloceren. Redenen dat files niet onafhankelijk van elkaar zijn te alloceren zijn het voorkomen van multi-file transacties met een meer ingewikkelde verwerkingswijze of restricties als capaciteiten en responstijden.
Als men fragmentering toestaat dan wordt de complexiteit nog groter.

2. GLOBAAL LITERATUUROVERZICHT.

Alle beschreven literatuur onderscheidt query en update transacties. In de hierna volgende tabel wordt aangegeven wat wel of niet waar wordt gemodelleerd of gedaan (X = wel). De meeste facetten zijn reeds onder 1. beschreven. Voor de overige kan worden opgemerkt:

- één file : geeft aan dat copieën van slechts één file worden gealloceerd.
- multi-file query : als naast X een N wordt vermeld, betekent dat dat hier niet een eenvoudige verwerkingswijze wordt verondersteld.
- niet vaste routing : geeft aan dat de route van de gegevens voor een bepaalde transactie gedetermineerd uit de file allocatie volgt; is dit niet zo, dan ligt meestal de verhouding van de verschillende mogelijke routeringen vast.
- VOLLEDIG NETWERK : geeft aan of het essentiëel voor het model is dat men een volledig netwerk veronderstelt.
- EXTRA'S : geeft aan of naast het pure file allocatie probleem ook andere aspecten mede worden geoptimaliseerd.
- SOORT PROBLEEM : geeft aan wat voor model uiteindelijk resulteert:
  - INLP = integer niet-lineair probleem.
  - ILP = integer lineair probleem.
  - MILP = mixed integer lineair probleem
- OPLOSSING : geeft aan of er optimale of heuristische algoritmen voor het probleem worden genoemd of beschreven, respectievelijk met een O en een H.

	1 CHU69	2 CHU73	3 CAS72	4 CHA76	5 MAH76	6 MOR77	7 RAM79	8 CHES0	9 FIS80	10 IRA81	11 APE81	12 CER82	13 CHA82	14 LAN83	15 RAM83	16 CER83
FILES EN TRANSACTIES																
één file			X	X											X	
progr.en datafile						X		X	X							
fragmentatie											X					X
multi-file query							X				XN		X			XN
KOSTEN																
transport	X	X	X	X		X	X	X	X	X	X		X	X	X	X
opslag	X	X	X	X	X	X	X	X	X	X				X	X	
andere					X[5i)]			X[8i)]				X[12)]	X[13)]			X[16i
RESTRICTIES																
opslagcapaciteit	X	X							X			X	X			X
kanaalcapaciteit					X			X								
siteverwerkingscap.								X								
responstijd	X	X			X									X		
filebeschikbaarheid		X			X									X		
ROUTING																
niet vaste routing	X	X			X	X						X	X	X		
heen- en terugverkeer					X						X					
VOLLEDIG NETWERK	X	X											X			X
EXTRA'S					X[5ii)]			X[8ii)]			X[11)]			X[14)]		X[16i
SOORT PROBLEEM	ILP	ILP	ILP	ILP	INLP	MIP	ILP	INLP	ILP	ILP	ILP	ILP	ILP	ILP	ILP	ILP
OPLOSSING	O	O	O H	H	H O	O	-	O	O H	H	O H	O H	O	H	H	H

Noten :

5i) Kosten per tijdseenheid voor het alloceren van een kanaal van een bepaalde capaciteit.

5ii) De capaciteiten van kanalen worden mede geoptimaliseerd.

8i) Deze kosten zijn per tijdseenheid :

- Kosten verbonden aan de keuze van een computer met een bepaalde verwerkingscapaciteit.
- Kosten voor het alloceren van een kanaal van een bepaalde capaciteit.

8ii) Zowel computer- als kanaalverwerkingscapaciteiten worden mede geoptimaliseerd.

11) Queryverwerking wordt tot op zekere hoogte mede geoptimaliseerd.

12) De kosten bestaan uit 'toegangskosten' (access costs) tot een file. Deze zijn lineair afhankelijk van het aantal toegangen en of een file elders of ter plekke geraadpleegd wordt.

13) De updatekosten worden hier gezien als concurrency control kosten en zijn een lineaire functie van het aantal copieën.

14) Het effect van alternatieve routeringen in geval van tijdelijke site- of kanaalonbeschikbaarheid wordt meegenomen m.b.t de performance criteria.

15i) Zie 12).

15ii) Partitionering van files wordt mede geoptimaliseerd.

3. GEDETAILLEERD LITERATUUROVERZICHT.

Afzonderlijk zullen nu de verschillende bijdragen worden samengevat met de nadruk op de wijze van modelleren en de manier van oplossen. Dit gebeurt op chronologische volgorde met uitzondering van bij elkaar horende bijdragen. Waar in het kader van een samenvatting dat mogelijk was, en waar speciale (bijv. tot nu toe niet bekeken of belangrijke) facetten voorkomen, is gepoogd de verschillende modellen zo volledig mogelijk te beschrijven. De aard van de modellen maakt het gebruik van heuristieken veelal nodig: daarom tot slot een kort overzicht van de belangrijkste heuristieken die worden gebruikt (pag.19).

Op de volgende pagina wordt een lijst van algemeen gebruikte variabelen en parameters gegeven; specifieke worden apart bij de bijdragen vermeld.

Lijst van algemeen gebruikte variabelen en parameters.

n : het aantal sites.

s,t,v : indices om de sites aan te duiden, $s,t,v=1,\ldots,n$.

C_s : de opslagcapaciteit in eenheden gegevens in site s.

m : het aantal verschillende files.

f : file index, $f=1,\ldots,m$.

m_f : het aantal copieën van file f.

L_f : de omvang van file f in eenheden gegevens.

o : het aantal programma's.

p : programmaindex, $p=1,\ldots,o$.

X : allocatievariabele, $X = \begin{cases} 1 \text{ als object aanwezig is.} \\ 0 \text{ anders.} \end{cases}$

$X = X_s$, in geval van slechts één file : geeft aan of file in site s aanwezig is.

$X = X_{sf}$, in geval van meer dan één file : geeft aan of file f in site s aanwezig is.

$X = X_s^p$, geeft aan of programma p in site s aanwezig is.

Y : transactieroutingvariabele.

$Y = Y_{st}$: de fractie van het queryverkeer ontstaan in s dat aan t wordt toegewezen (in geval van één file).

$Y = Y_{tfv}$: de fractie van het queryverkeer voor f vanuit t dat aan v wordt toegewezen (in geval van meer dan één file).

$Y = Y_{sft}^p$: de fractie van het verkeer voor p vanuit s dat aan t wordt toegewezen en dat file f raadpleegt.

λ : de hoeveelheid queryverkeer voor een file in eenheden gegevens per tijdseenheid.

$\lambda = \lambda_s$: vanuit s (in geval van één file).

$\lambda = \lambda_{sf}$: vanuit s voor file f.

$\lambda = \lambda_{sf}^p$: vanuit s voor file f dat loopt via p.

$\psi_s, \psi_{sf}, \psi_{sf}^p$: analoog aan λ voor updateverkeer.

σ : de opslagkosten per tijdseenheid per eenheid gegeven.

$\sigma = \sigma_s$ als deze alleen afhankelijk zijn van de site.

$\sigma = \sigma_{sf}$ of σ_{sp} als deze ook nog afhankelijk zijn van het object.

τ_{st} : de transportkosten per eenheid verkeer per tijdseenheid tussen s en t, evt. $\tau_{st}^{(u)}$ of $\tau_{st}^{(q)}$ voor resp. update- of queryverkeer.

1,2. [CHU69],[CHU73]

Chu [CHU69] was de eerste die het file allocatie probleem aan de orde stelde, en wel voor een multi-computer systeem.

Speciale aannamen: Per file worden een vast aantal copieën gealloceerd. Het verkeer tussen twee sites wordt als een M|D|1-rij gemodelleerd, onder aanname dat het netwerk volledig is. Queryverkeer van een bepaalde file wordt gelijkelijk over de copieën verdeeld indien een copie niet in de host-site aanwezig is.

Model: [CHU73]

A_s : de kans dat site s werkt.

A_{st} : de kans dat kanaal tussen s en t werkt.

A^f : de minimaal gewenste kans dat file f beschikbaar is.

T_{sf}^{max} : de maximaal toegestane gemiddelde responstijd als file f vanuit s wordt geraadpleegd.

T_{stf} : de gemiddelde responstijd als file f in t vanuit s wordt geraadpleegd.

Het probleem luidt nu:

minimaliseer :

$$\underset{s}{\Sigma}\underset{f}{\Sigma}\,\sigma_{sf}\cdot X_{sf} + \underset{s}{\Sigma}\underset{t}{\Sigma}\underset{f}{\Sigma}\,\frac{1}{m_f}\cdot\lambda_{sf}\tau_{sf}\cdot X_{tf}(1-X_{sf}) + \underset{s}{\Sigma}\underset{t}{\Sigma}\underset{f}{\Sigma}\,\psi_{sf}\tau_{st}\cdot X_{tf}$$

(opslag) (query) (update)

onder voorwaarde van :

$$\underset{s}{\Sigma} X_{sf} = m_f \quad ,\forall_f \qquad \{\text{alle copieën gealloceerd}\}$$

$$\underset{f}{\Sigma} L_f \cdot X_{sf} \leq C_s \quad ,\forall_s \qquad \{\text{opslagcapaciteit}\}$$

$$(1-X_{sf})\cdot X_{tf}\cdot T_{stf} \leq T_{sf}^{max} \quad ,\forall_{s,t,f} \qquad \{\text{responstijd}\}$$

($T_{stf} \approx$ een functie van X en de modelparameters')

$$A_s(1-(1-A_sA_{st})^{m_f}) \geq A^f \quad ,\forall_f \qquad \{\text{filebeschikbaarheid}\}$$

$$X_{sf} = 0 \mid 1 \quad ,\forall_{s,f} \qquad \{\text{integer voorwaarde}\}$$

Opm.: σ, λ en ψ zijn produkten van gedetailleerdere modelparameters.

Oplossing : Dit is een niet-lineair integer probleem, dat door het toevoegen van extra variabelen en restricties kan worden getransformeerd tot een integer lineair probleem. Aanbevolen wordt Gomory's cutting plane techniek die een optimale oplossing bepaalt, echter voor grote problemen loopt de benodigde rekentijd uit de hand.

3,15. [CAS72],[RAM83]

Casey [CAS72] benadrukte de rol van de verhouding tussen query- en update-verkeer op de file allocatie.

Speciale aanname: Er wordt slechts één file beschouwd, zonder verdere restricties.

Model:

minimaliseer :

$$\underset{s}{\Sigma}\,\underset{t}{\Sigma}\,\psi_s \tau_{st}^{(u)} \cdot X_t + \underset{s}{\Sigma}\,\lambda_s \cdot \min_{t, X_{tf}=1}\{\tau_{st}^{(q)}\} + \underset{s}{\Sigma}\,\sigma_s \cdot X_s$$

{update} {query} {opslag}

onder voorwaarde van :

$$X_s = 0 \mid 1 \quad ,\forall_s \qquad \{\text{integer voorwaarde}\}$$

Opm.: Casey wijst op de mogelijk verstandiger routing van update verkeer in geval van meer dan één copie: in plaats van dat elke copie direct vanuit de host-site een update boodschap krijgt, kunnen deze boodschappen eventueel over deels gemeenschappelijke routes, opdat de kosten minder zijn.

Oplossing : Casey beschrijft een enumeratie-techniek door paden in de zgn. cost-graph (zie als voorbeeld fig.1 voor n=3) efficiënt te doorlopen en daarnaast heuristische technieken zoals een aanpak waarbij van bovengenoemde paden alleen de meestbelovende worden doorlopen. Eswaran [ESW74] bewees dat dit probleem NP-volledig is, en toonde daarmee het belang van heuristische benaderingen aan. Grapa en Belford [GRA77] bepaalden eenvoudige voorwaarden die vooraf al met zekerheid aan een aantal variabelen de juiste waarde geven. Ramamoorthy en Wah [RAM83] toonden de isomorfie aan tussen dit probleem en het zgn. 'single commodity warehouse location' probleem (scwlp), een bekend probleem uit de operations research. Met behulp van [GRA77] en voorwaarden uit [EFR66] voor het scwlp, wordt een heuristiek geformuleerd: De variabelen worden in drie sets verdeeld:

$K_0 = \{s \mid X_s=0\}$; $K_1 = \{s \mid X_s=1\}$ en $K_2 = \{s \mid X_s$ heeft nog geen waarde$\}$. Iteratief wordt nu steeds de meest belovende variabele uit K_2 geselecteerd en aan K_0 of K_1 toegevoegd, totdat $K_2 = \emptyset$.

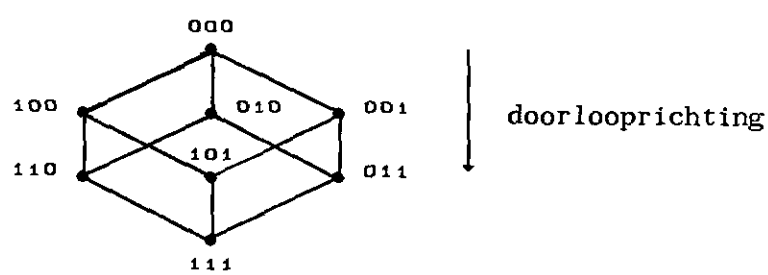

fig.1
cost-graph,n=3

4. [CHA76]

Chandy en Hewes formuleren een soortgelijk probleem als Casey.

Speciale aanname: De updatekosten worden geïncorporeerd in de opslagkosten, er zijn dus geen update-transportkosten.

Model:

minimaliseer: $\underset{\{\text{opslag/update}\}}{\sum_s \sigma_s \cdot X_s} + \underset{\{\text{query}\}}{\sum_s \sum_{t \neq s} \lambda_s \tau_{st} \cdot Y_{st}}$

onder voorwaarde van :

$$\sum_{t \neq s} Y_{st} + X_s = 1 \quad ,\forall_s \qquad \{\text{query aan één site toegewezen, zo mogelijk aan de host-site}\}$$

$$Y_{st} \leq X_t \quad ,\forall_{s \neq t} \qquad \{\text{query alleen toewijzen aan aanwezige file}\}$$

$$Y_{st}, X_s = 0 \mid 1 \quad ,\forall_{s,t} \qquad \{\text{integer voorwaarde}\}$$

Opm.: We zien hier voor het eerst het gebruik van transactietoewijzingsvariabelen.

Oplossing : Een manier van oplossen is de LP-relaxatie ($X_s, Y_{st} = 0 \mid 1$ vervangen door $X_s, Y_{st} \geq 0$). Oplossingen hiervan zijn mogelijk niet toegelaten, omdat files niet mogen worden gepartitioneerd. In veel gevallen geeft de LP-oplossing automatisch een geheeltallige oplossing. Daarnaast verschaft die een ondergrens waaraan de kwaliteit van een heuristische oplossing kan worden gespiegeld. Een eenvoudige heuristiek luidt : Ken initiëel één file toe aan een of andere site (start-allocatie); Kies dan per iteratie voor de goedkoopste allocatie die binnen afstand d van de vorige allocatie ligt (dwz. dat ten hoogste d copieën worden toegevoegd of weggelaten uit de vorige allocatie) ; Ga zo door totdat geen kostenreductie meer mogelijk is. Een keuze van d=2 blijkt goed te werken.

5. [MAH76]

Mahmoud en Riordan beschouwen het file allocatie probleem in samenhang met de optimale toekenning van capaciteiten aan de kanalen.

Speciale aannamen: Zowel heen- als terugverkeer voor queries en updates ; queryverkeer voor een bepaalde file wordt gelijkelijk over de copieën verdeeld als geen copie in de host-site aanwezig is ; het verkeer door een kanaal wordt als een M|M|1-rij beschouwd; géén transportkosten.

Model: De benaderde niet-lineaire uitdrukkingen voor responstijden en filebeschikbaarheden geven met de lineaire doelfunctie een niet-lineair integer probleem.

Oplossing : Een heuristische procedure wordt beschreven die uit een start- en een optimaliseringsroutine bestaat. De startroutine vindt een verzameling toegelaten oplossingen door het minimum aantal copieën per file te bepalen (uit de filebeschikbaarheidsrestrictie) en te alloceren. Per allocatie worden de capaciteiten van de kanalen geoptimaliseerd. In de optimaliseringsroutine worden goedkopere allocaties gezocht door middel van een add/delete (van een copie) routine; capaciteiten worden geheroptimaliseerd. Dit gaat zo door totdat geen kostenreduktie meer mogelijk is.

6,9. [MOR77],[FIS80]

Morgan en Levin [MOR77] maken onderscheid tussen programma- en datafiles. Zij stellen dat een transactie eerst door een programma moet worden verwerkt voordat de datafile kan worden geraadpleegd. Bovendien zijn mogelijk de programma's niet overal te situeren.

Speciale aannamen: Programma- en datafiles; Alléén heenverkeer voor zowel queries als updates.

Model: J^p: de verzameling sites waar programma p gealloceerd kan worden.

$\alpha^{(u)}, \alpha^{(q)}$: de expansiefactor voor update- resp. queryverkeer, dwz. als V het verkeer naar p is,dan is het verkeer naar de betreffende file f,αV.

Het probleem luidt nu:

minimaliseer:

$$\underset{s\,t\,f\,p}{\Sigma\Sigma\Sigma\Sigma} \lambda^p_{sf}\tau^{(q)}_{st}\cdot Y^p_{sft} \quad + \quad \underset{s\,t\,f\,p}{\Sigma\Sigma\Sigma\Sigma} \psi^p_{sf}\tau^{(u)}_{st}\cdot Y^p_{sft} +$$

(pr.file query) (pr.file update)

$$+ \underset{t\,p}{\Sigma\Sigma} \sigma^p_t\cdot X^p_t \quad + \quad \underset{v\,f}{\Sigma\Sigma} \sigma_{vf}\cdot X_{vft} +$$

(pr.file opslag) (dt.file opslag)

$$+ \underset{s\,t\,v\,f\,p}{\Sigma\Sigma\Sigma\Sigma\Sigma} \alpha^{(q)}\lambda^p_{sf}\tau^{(q)}_{sv}\cdot Y^p_{sft}\cdot Y_{tfv} + \underset{s\,t\,v\,f\,p}{\Sigma\Sigma\Sigma\Sigma\Sigma} \alpha^{(u)}\psi^p_{sf}\tau^{(u)}_{sv}\cdot Y^p_{sft}\cdot X_{vf}$$

(dt.file query) (dt.file update)

onder voorwaarde van :

$$\underset{t}{\Sigma} X^p_t \geq 1 \text{ en } \underset{v}{\Sigma} X_{vf} \geq 1 \qquad \{\text{minimaal één copie}\}$$

$$\underset{t}{\Sigma} Y^p_{sft} \geq 1 \text{ en } \underset{v}{\Sigma} Y_{tfv} \geq 1 \qquad \{\text{alle verkeer toegewezen}\}$$

$$\underset{s}{\Sigma} Y^p_{sft} \leq n\cdot X^p_t \text{ en } \underset{t}{\Sigma} Y_{tfv} \leq n\cdot X_{vf} \qquad \{\text{verkeer alleen langs aanwezige files}\}$$

$$X^p_t = 0 \quad ,\forall_{t\notin J^p} \qquad \{\text{pr.files alleen in bepaalde sites}\}$$

$$0 \leq Y^p_{sft} \,,\, Y_{tfv} \leq 1 \qquad \{\text{Y's zijn fracties}\}$$

X^{p}_{t} , X_{vf} = 0 | 1 {integer voorwaarde}

Opm.: Het toelaten van gefractioneerde transactietoewijzingsvariabelen staat toe dat men later op basis van andere, niet gemodelleerde criteria nu eens voor de ene en dan weer voor de andere routing kiest.

Oplossing : Als programmafile-opslagkosten worden verwaarloosd, kan het probleem worden herleid tot individuele file allocatie problemen.

Een enumeratie-algoritme met behulp van hypercubi wordt beschreven. Dit is een 'meerdimensionale' variant op Casey's cost-graphs.

Fisher en Hochbaum [FIS80] breiden het model verder uit met opslagcapaciteiten en geven een lineair model. Zij beschrijven efficiëntere algoritmen zoals: een optimale branch-and-bound methode; een greedy heuristiek die steeds één copie van een file toevoegt; een interchange heuristiek die ook verwisselingen van een copie van de ene naar de andere site toestaat; en de Lagrange-relaxatie techniek.

7. [RAM79]

Bij Ramamoorthy en Wah treffen we als eerste multi-file transacties aan en wel in een relationele database omgeving. Aangetoond wordt dat voor zgn. decomponeerbare queries, dat zijn queries waarbij het niet noodzakelijk is dat (delen van) twee (of meer) relaties per se in de host-site aanwezig zijn wil de query uit te voeren zijn, het probleem herleid kan worden tot m onafhankelijke problemen voor het alloceren van één file (relatie). Dit is mogelijk door de eenvoudige verwerkingswijze en het niet beschouwen van capaciteiten en performance criteria.

Voor de niet-decomponeerbare queries wordt een techniek beschreven om deze decomponeerbaar te maken door het toevoegen van redundante informatie aan sommige relaties. Dit kan leiden tot een stijging van updatekosten maar tot een grotere daling van querykosten.

8. [CHE80]

Chen en Akoka optimaliseren naast de allocatie van (programma- en data-) files ook de toekenning van kanaal- en siteverwerkingscapaciteiten.

Speciale aannamen: Programma- en datafiles; Zowel heen- als terugverkeer.

Model: Er resulteert een omvangrijk niet-lineair integer probleem, met zowel in de doelfunctie als in de restricties niet-lineairiteiten. Bovendien is de doelfunctie niet convex.

Oplossing : Vanwege de niet-convexe doelfunctie wordt een heuristische benadering afgewezen (hoewel het aantal variabelen en restricties daar wel

aanleiding toe geeft). Een optimale oplossing wordt verkregen door een bekende branch-and-bound methode voor mixed-integer problemen toe te passen en daarbij gebruik te maken van de structuur van het probleem.

10. [IRA81]

Irani en Khabbaz optimaliseren de topologie van het netwerk en de kanaalcapaciteiten in samenhang met de file allocatie. Naast de gebruikelijke kosten worden performance criteria als filebeschikbaarheden en responstijden beschouwd.

Speciale aanname: Een kanaal wordt als een M|M|1-rij beschouwd.

Model: Er resteert een integer probleem.

Oplossing : Het oplossen van het probleem is een iteratieproces rond de parameter κ, de netwerkconnectiviteit. Vooraf kan een ondergrens van κ bepaald worden op basis van een netwerkbetrouwbaarheidsrestrictie. Deze houdt in dat de kans dat het netwerk op een willekeurig moment verbonden is, een bepaald minimum moet overschrijden. Tijdens het itereren rond κ kan worden vastgesteld dat verder ophogen niet meer zinvol is. Gegeven κ, worden nu achtereenvolgens gescheiden bepaald (waarbij de uitkomst van een vorig probleem input vormt voor het opvolgende probleem):

1. een optimale topologie, beperkt tot een verzameling (vooraf) toegestane topologiën, door middel van respectievelijk een greedy en interchange heuristiek.
2. een optimale file allocatie.
   Dit gebeurt per file afzonderlijk en in een iteratie rond m_f, het aantal copieën van f. Op grond van de filebeschikbaarheid is dit aan een minimum gebonden. m_f copieën worden dan achtereenvolgens gealloceerd door een greedy en een interchange heuristiek.
3. de kanaalcapaciteitsallocatie.
   Gegeven de topologie uit 1. en de file allocatie uit 2. worden nu de kanaalcapaciteitskosten geminimaliseerd onder voorwaarde dat de gemiddelde responstijden acceptabel blijven. Dit gebeurt optimaal.

11. [APE81]

Apers pakt het file allocatie probleem tegelijk aan met partitionering (fragmentatie) van de files en met query-optimalisering. Files worden beschouwd als relaties, de te alloceren objecten zijn nu fragmenten van relaties en wel horizontaal 'gesplit' op basis van de 'clauses' in de queries en updates en verticaal op basis van de attributen (4.3.2).

Speciale aannamen: Transportkosten enkel afhankelijk van hoeveelheid gegevens, dus onafhankelijk van het kanaal; Verder geen restricties beschouwd.

Model: De volgende begrippen worden geïntroduceerd:

nucleus-site (NS): een 2-tupel (F,O) met F een verzameling fragmenten en O een verzameling operaties die resp. in NS liggen en worden uitgevoerd. Een NS kan zijn:

physical-site (PhS): een echte site in het netwerk, of een

virtual-site (VS): een fictieve site.

Een NS kan een aantal NS's aan zich hebben toegekend. Dat betekent dat er een datatransport van deze laatsten naar de eerste plaatsvindt. Een VS kan aan ten hoogste één NS toegekend zijn, een PhS aan geen enkele. Een 'verbinding' tussen twee NS's is een NS die als F en O de vereniging van de twee F's en O's heeft. Mogelijk zijn:

Een VS_1 verbinden met een VS_2 levert een nieuwe VS, en een VS verbinden met een PhS levert een nieuwe PhS. Zo'n verbinding betekent dat (uiteindelijk) de fragmenten en operaties in dezelfde echte site zullen liggen.

Er zijn drie soorten allocaties:

initial allocation (ia): $\forall_{VS}[\ |F_{VS}| \leq 1\ ]$ en $\forall_{PhS}[\ F_{PhS} = \emptyset\ ]$ en $\forall_{VS}\forall_{PhS}[$ VS niet toegekend aan PhS $]$.

partially specified allocation (psa):

$\exists_{VS}[\ \exists_{NS}[$VS toegekend aan of verbonden met NS $]]$.

completely specified allocation (csa): $\forall_{VS}\exists_{PhS}[$ VS verbonden met PhS $]$.

Om de kosten van een allocatie te bepalen wordt een processing schedules graph (psg) bepaald. Deze bestaat uit PhS-knopen, VS-knopen en gerichte takken, elk met een label (i,λ,d), waarbij i de transactie, λ de frequentie en de hoeveelheid getransporteerde gegevens is.

Een psg wordt nu als volgt bepaald: laat op de transacties en een allocatie een query processing algoritme (qpa) los: dit levert een schedule ofwel VS's die aan elkaar zijn toegekend. Het minimaliseren van de transportkosten komt nu neer op het verwijderen van takken uit de psg door VS's met elkaar of met een PhS te verbinden totdat een optimale csa ontstaat.

Er zijn nu twee benaderingen mogelijk resp. een statische en een dynamische. In het eerste geval blijft een eenmaal bepaalde schedule vast om een csa te

bepalen, in het tweede geval wordt met het wijzigen van de allocatie (ofwel de psg) een nieuwe schedule bepaald door het qpa.
Voor statische schedules is het probleem als volgt te formuleren:
minimaliseer :

$$\sum_{i,j}\sum_{p,q} c_{ijpq} \cdot X_{ij} \cdot X_{pq} + \sum_{i,j} b_{ij} \cdot (1-X_{ij})$$

onder voorwaarde van :

$$X_{ij} = 0 \mid 1$$

met:

$X_{ij} = \begin{cases} 1 & \text{als } VS_i \text{ verbonden met } PhS_j . \\ 0 & \text{anders}. \end{cases}$

$c_{ijpq} = \begin{cases} \text{de totale hoeveelheid data getransporteerd tussen } VS_i \text{ en } VS_p, & j \neq q . \\ 0 & j \neq q. \end{cases}$

b_{ij} = de totale hoeveelheid data getransporteerd tussen VS_i en PhS_j .

Dit is een quadratic assignment-achtig probleem.
Oplossing : Standaard integer programmeringstechnieken kunnen worden toegepast. Apers geeft een schatter die bij het toepassen van het Heuristische Pad Algoritme (HPA) een optimale oplossing van dit probleem oplevert (psa static cost, p.125). Een heuristische allocatie wordt gevonden door het algoritme 'total data allocation' (p.129). Uitgaande van de psg wordt voor twee nucleus-sites NS_i en NS_j $LINK_{ij}$ gedefiniëerd: Dit is de som van de labels die verdwijnen als de twee worden verbonden of als de een aan de ander wordt toegekend. Initiëel wordt gestart met een psa waarbij elke VS_i toegekend is aan de PhS_j waarvoor $LINK_{ij}$ maximaal is. Iteratief worden nu steeds die twee VS's verbonden waarvoor $LINK_{ij}$ maximaal is (en >0). Dit gaat zo door totdat geen VS's meer verbonden kunnen worden of totdat de maximale $LINK_{ij}$ kleiner dan nul is. Voor bepaalde zgn. 'simple' psg's is dit algoritme nog optimaal. Ten opzichte van het HPA levert het gemiddeld zo'n 3% hogere kosten.
Het gebruik van dynamische schedules tijdens het optimaliseringsproces kan dit proces te rekenintensief maken. Onder de aanname dat elke query een relatief klein aantal fragmenten raadpleegt, kan een efficiënte schatter worden gegeven om te gebruiken in het HPA (psa dynamic cost,p.138). Dit levert dan weer een optimale oplossing.

12. [CER82]

Ceri et al. onderscheiden meerdere files en een opslagcapaciteit per site. De kosten zijn de per eenheid kosten van access op een file, maar zijn ook te interpreteren als transportkosten. De kosten zijn afhankelijk van of er een locale of een 'remote' access is. Transacties zijn multi-file transacties en bestaan uit een aantal retrieval (query) en update accesses.

Model:

nt : het aantal transacties.
k : transactie index, k = 1,...,nt.
φ_{ks} : frekwentie van transactie k met s als host-site.
n_{kf} : het aantal retrieval accesses van k op file f.
n'_{kf} : het aantal update accesses van k op file f.
c_{loc} : per eenheid kosten voor een locale access.
c_{rem} : per eenheid kosten voor een remote access.

Het probleem luidt nu:

minimaliseer :

$$\sum_s \sum_f \sum_k \varphi_{ks} n_{kf} \cdot [c_{loc} \cdot X_{sf} + c_{rem} \cdot (1-X_{sf})] + \sum_s \sum_f \sum_k \varphi_{ks} n'_{kf} c_{loc} \cdot X_{sf} +$$

(retrievals) (locale updates)

$$+ \sum_s \sum_f \sum_k \{X_{sf} \cdot [\sum_{t \neq s} \varphi_{kt} n'_{kf} c_{rem}]\}$$

(remote updates)

onder voorwaarde van :

$$\sum_f L_f \cdot X_{sf} \leq C_s \quad ,\forall_f \qquad \{\text{opslagcapaciteit}\}$$
$$\sum_s X_{sf} = 1 \quad ,\forall_f \qquad \{\text{precies één copie}\} \quad (1)$$

of:

$$\sum_s X_{sf} \geq 1 \quad ,\forall_f \qquad \{\text{minimaal één copie}\} \quad (2)$$
$$X_{sf} = 0 \mid 1 \quad ,\forall_{s,f}$$

Oplossing :Het probleem is, zowel voor geval (1) als voor (2), uiteindelijk als een maximaliseringsprobleem te formuleren met als doelfunctie :

$\sum_s \sum_f v_{sf} \cdot X_{sf}$, met v_{sf} een expressie van de genoemde parameters.

Geval (1) is dan een zgn. multiple choice constraint knapsack probleem, waarvoor speciale algoritmen bekend zijn (zie ref.[18]-[24] in [CER82]). Geval (2) wordt door een branch-and-bound methode opgelost die bestaat uit het decomponeren van het probleem tot vele 'simple knapsack'-problemen. Afhankelijk van de wijze waarop deze worden opgelost, verkrijgt men een optimaal of heuristisch algoritme.

13. [CHA82]

Chang en Liu zien een query als een view, dat is een deelverzameling van de verzameling files. Per site wordt nu zo'n view met een bepaalde kans gewenst. Een file wordt met een bepaalde kans geupdated. De kosten die daaraan verbonden zijn, worden gezien als concurrency control kosten en zijn een lineaire functie van het aantal copieën. Per view worden hele files naar de host-site verzonden.

Speciale aannamen : Geen transportkosten voor updates; de transportkosten zijn gelijk voor elk kanaal.

Model: Het probleem dat uiteindelijk resteert luidt :

minimaliseer :

$$\sum_s \sum_f w_{sf} \cdot X_{sf}$$

onder voorwaarde van:

$$\sum_f X_{sf} \leq n^s \quad ,\forall_s \quad \{\text{max. } n^s \text{ files in site s}\}$$

$$\sum_s X_{sf} \leq m^f \quad ,\forall_f \quad \{\text{max. } m^f \text{ files van file f}\}$$

$$\sum_s X_{sf} \geq 1 \quad ,\forall_f \quad \{\text{minimaal één copie van f}\}$$

$$X_{sf} = 0 \mid 1 \quad ,\forall_{s,f}$$

De term w_{sf} is een expressie van de genoemde kansen, filegroottes en kosten.

Oplossing : Het probleem wordt getransformeerd tot een minimum cost network flow probleem en opgelost met het zgn. out-of-kilter algoritme dat voor grotere problemen essentiëel efficiënter blijkt te zijn dan een algemeen depth-first branch-and-bound algoritme, echter nog steeds (te) rekenintensief voor echt grote problemen.

14. [LAN83]

Laning en Leonard beschouwen het file allocatie probleem in netwerken die mogelijkheden hebben tot 'adaptieve' routering. Initiële kandidaat file allocaties worden bepaald door per file iteratief het zgn. p-mediaan probleem op te lossen (met $p = m_f$) op grond van transport en opslagkosten. Vervolgens worden door middel van simulatie en een aantal fileverplaatsingsregels hieruit (suboptimale) allocaties bepaald die aan de filebeschikbaarheids- en responstijdrestricties voldoen.

16. [CER83]

Ceri et al. stellen de volgende methodologie voor om files in een gedistribueerde omgeving te alloceren :

1) Specificeer kandidaat partitioneringen, vervolgens :
2) Bepaal de beste partitie tegelijk met de niet-gerepliceerde allocatie van de (gefragmenteerde) files.
3) Repliceer vanuit deze allocatie de fragmenten om tot verdere kostenreductie te komen.

De database wordt gemodelleerd als een logisch schema in de vorm van een gerichte graaf met de files als knopen en links tussen files als takken. Een link staat voor een 1:n relatie tussen twee files en levert een 'owner' en 'member' van die link. Een transactie is nu een pad in de graaf. Naast transportkosten,gelijk voor elk kanaal, worden verwerkingskosten beschouwd,en wel 'access'-kosten.

Niet duidelijk wordt aangegeven hoe men tot kandidaat partitioneringen komt: men stelt dat deze uit de 'clauses' in de transacties kunnen komen alsmede uit de frequentie van de transacties.

Voor 2) wordt nu het volgende geformuleerd :

Laat p en h resp. de partitie- en linkindex zijn, dan wordt gedefiniëerd:

$$S_{fp} = \begin{cases} 1 \text{ als partitie p voor f gekozen is.} \\ 0 \text{ anders.} \end{cases}$$

$$X_{sf} = \begin{cases} 1 \text{ als f in zijn geheel in s gealloceerd is.} \\ 0 \text{ anders.} \end{cases}$$

$$W_{hp} = \begin{cases} 1 \text{ als link h gebruikt is om een partitie af te leiden in de hiërarchie van afgeleide partitities van partitionering p en owner en member van h beide volgens p zijn gepartitioneerd.} \\ 0 \text{ anders.} \end{cases}$$

$$V_{hs} = \begin{cases} 1 \text{ als link h 'lokaal' is t.o.v. site s omdat owner en member van h geheel in s zijn gealloceerd.} \\ 0 \text{ anders.} \end{cases}$$

Summier is nu het volgende integer LP-probleem te formuleren:

minimaliseer:

$$\sum_{f,p} c_{fp} \cdot S_{fp} + \sum_{s,f} d_{sf} \cdot X_{sf} - \sum_{h,p} a_{hp} \cdot W_{hp} - \sum_{h,s} b_h \cdot V_{hs}$$

onder voorwaarde van:

i) $\sum_{p} S_{fp} + \sum_{s} X_{sf} = 1 \quad ,\forall_f$ {precies één (gepartitioneerde) copie}

ii) + een aantal consistentie- en afhankelijkheidsrestricties.

c_{fp} en d_{sf} zijn de transport- en verwerkingskosten gerelateerd aan de betreffende beslissingen over S en X, a_{hp} en b_h zijn de transportkosten die bespaard kunnen worden doordat resp. dezelfde partitie is gebruikt of member en owner

in dezelfde site zijn opgeslagen. Deze coëfficiënten zijn (ingewikkelde) functies van andere gedetailleerde modelparameters.
Het probleem kan een te grote complexiteit hebben. Daarom wordt een decompositiemodel geformuleerd om een aantal subproblemen te krijgen van een begrensde complexiteit. Zo'n subprobleem beschouwt slechts een deel van de files en van de links. 'Optimale' subproblemen worden bepaald door het minimaliseren van transportkosten langs links tussen files die tot verschillende subproblemen behoren.
Een oplossing van 3) wordt bepaald door een greedy heuristiek met als startpunt de allocatie uit 2). Per iteratie wordt nu dat fragment gerepliceerd, waardoor er in die iteratie de grootste kostenreductie plaatsvindt.

Heuristieken.

Tot slot een opsomming van de belangrijkste heuristieken die bij het file allocatie probleem voorkomen:
1) LP-relaxatie. De geheeltalligheidseis laat men in eerste instantie vallen om een evt. gebroken oplossing daarna te veranderen in een in de buurt gelegen toegelaten gehele oplossing ([CHA76]).
2) Een 'één-staps zoek-heuristiek'. Dit zijn heuristieken die per iteratie een toelaten allocatie vinden door de vorige op een bepaalde manier te veranderen, bijv. door het toevoegen, verwijderen of verplaatsen van een file (resp. zgn. add- , delete- en interchangeroutines), zodanig dat de nieuwe allocatie beter is. Een greedy heuristiek is een heuristiek die in zo'n iteratie steeds voor die beschouwde verandering kiest die in die iteratie de meeste kostenreductie oplevert ([CAS72],[MAH76],[FIS80],[IRA81],[APE81],[RAM83] en [CER83]).
3) Een meerstaps heuristiek. Dit is een uitbreiding van 2) maar nu is per iteratie meer dan één verandering toegestaan (niet meer dan een vast aantal), zodanig dat tussentijds in een iteratie een verandering kostenverhogend kan zijn, maar de totale veranderingen in een iteratie zijn tesamen kostenverlagend ([CHA76]).

4. OP WEG NAAR EEN OVERKOEPELEND MODEL.

In dit hoofdstuk zal een eigen file allocatie probleem worden besproken. De indeling is als volgt:

4.1 Modelformulering.

We nemen aan dat de te alloceren objecten al op een verstandige manier zijn gekozen (bijv. een op een bepaalde wijze gefragmenteerde verzameling files). We duiden de objecten verder aan met files. Het netwerk wordt volledig verondersteld. De manier van verwerken is die in de inleiding (zie hfst1. pag.2) geschetste eenvoudige wijze van verwerken:
Een query raadpleegt een aantal files. Naar een copie van een betreffende file gaat een bericht welke gegevens voor die query gevraagd worden. Het antwoord daarop wordt teruggezonden naar de host-site. Als een copie in de host-site aanwezig is, wordt het antwoord direct en zonder kosten beschikbaar gesteld. De omvang van een bericht wordt verwaarloosd ten opzichte van die van het antwoord. De antwoorden van de verschillende files worden in de host-site kosteloos verwerkt tot het gevraagde resultaat.
Een update gaat als volgt: Van de host-site naar elke copie van de betreffende file gaat een update-boodschap van een bepaalde omvang over het kanaal tussen de twee betreffende sites. Een eventuele bevestigingsboodschap wordt verwaarloosbaar geacht.
We voeren de volgende grootheden en parameters in (zie ook hfst.3 pag.7) :

Q : de verzameling queries q.

U : de verzameling updates u.

$r_q \in \mathbb{R}^m$: $r_q(f) = \begin{cases} 1 & \text{als q een beroep doet op file f} \\ 0 & \text{anders} \end{cases}$, $q \in Q$.

φ_q , φ_u : het gemiddeld aantal keren per tijdseenheid dat q resp. u wordt uitgevoerd, $q \in Q$ resp. $u \in U$.

$h(q)$, $h(u)$: de host-site van q ($\in Q$) resp. u ($\in U$).

$f(u)$: de file die door u ($\in U$) wordt 'geupdated' (updates worden als atomaire updates gezien).

$\bar{\sigma}_s$: de opslagkosten per eenheid van gegevens per tijdseenheid in site s (s = 1,...,n).

l_{qf} : de gemiddelde omvang van een antwoord op query q op file f, q ϵ Q en f = 1,...,m.

l'_u : de gemiddelde omvang van een update-boodschap van update u (ϵ U) voor file f(u).

CC_{st} : de maximale hoeveelheid eenheden gegevens die per tijdseenheid van site s naar site t kan worden verzonden, s,t = 1,...,n .

Als extra variabele naast X_{sf} wordt ingevoerd :

Y_{qsf} = de fractie van het aantal keren dat query q bij uitvoering aan site s wordt toegewezen met betrekking tot file f, q ϵ Q,s=1,...,n f=1,...,m ($0 \leq Y_{qsf} \leq 1$).

Het voordeel om per query een toewijzingsvariabele in te voeren in tegenstelling tot een geaggregeerde queryverkeer-toewijzingsvariabele Y (zie pag.7) is dat met grotere nauwkeurigheid schattingen voor de gemiddelde responstijden op queries kunnen worden bepaald. Dit is belangrijker als er (enkele) queries zijn die snel een antwoord behoeven.

4.2 Model I.

We definiëren nu (voor q ϵ Q, u ϵ U, s = 1,...,n en f = 1,...,m):

$\sigma_{sf} := \bar{\sigma}_f \cdot L_f$; $\bar{\tau}_{qsf} := \varphi_q \cdot l_{qf} \cdot r_q(f) \cdot \tau_{s,h(q)}$; $\lambda_{qf} := \varphi_q \cdot l_{qf} \cdot r_q(f)$ en :

$\tau_u := \varphi_u \cdot l'_u$; $\upsilon_{us} := \tau_u \cdot \tau_{h(u),s}$.

We formuleren nu het volgende mixed-integer LP-probleem (I) :

minimaliseer :

$$\underset{\text{(opslag)}}{\sum_s \sum_f \sigma_{sf} \cdot X_{sf}} + \underset{\text{(update)}}{\sum_u \sum_s \upsilon_{us} \cdot X_{s,f(u)}} + \underset{\text{(query)}}{\sum_q \sum_s \sum_f \bar{\tau}_{qsf} \cdot Y_{qsf}}$$

onder voorwaarde van :

(1) $\sum_s X_{sf} \geq 1$, $\forall_f$ {minimaal één copie}

(2) $\sum_s Y_{qsf} = 1$, $\forall_{q,f}[r_q(f)=1]$ {queries volledig toegewezen}

(3) $\sum_f L_f \cdot X_{sf} \leq C_s$, $\forall_s$ {opslagcapaciteit}

(4) $X_{h(q),f} = Y_{q,h(q),f}$, $\forall_{q,f}[r_q(f)=1]$ { query zo mogelijk aan host-site toewijzen }

(5) $\sum_{q,h(q)=t} \sum_f \lambda_{qf} \cdot Y_{qsf} + \sum_{u,h(u)=s} \tau_u \cdot X_{t,f(u)} \leq CC_{st}$, $\forall_{s,t}[s \neq t]$ {kanaalcapaciteit}

(6) $X_{sf} = 0 \mid 1$; $Y_{qsf} \geq 0$, $\forall_{q,s,f}[r_q(f)=1]$

4.2.1 Oplossing.

De oplossing van (I) is te verkrijgen door standaardtechnieken voor dit soort problemen toe te passen, die echter bij een groot aantal variabelen te veel rekentijd kunnen kosten. Een andere mogelijkheid is de geheeltalligheids-eis ten aanzien van de X-variabelen te relaxeren tot $0 \leq X \leq 1$: Dan ontstaat een LP-probleem dat ook bij grotere problemen efficiënt is op te lossen. In principe kan dit gebroken, dus niet toegelaten oplossingen opleveren. Eventueel is X_{sf} dan te interpreteren als de fractie van file f in site s. Het probleem is dan hoe de queries aan deze fracties moeten worden toegewezen, of dat men de queries wel aan de fracties kan toewijzen : de fracties bevatten mogelijk niet voldoende van de file voor een daarop betreffende query.

Een andere manier is de eventueel gebroken oplossing 'verstandig' af te ronden tot een toegelaten oplossing :

Stel : X_{sf} gebroken, dan zijn er twee mogelijkheden :

$\underline{1}$ X_{sf} wordt 0 , of $\underline{2}$ X_{sf} wordt 1 .

Gevolgen hiervan zijn:

$\underline{1}$: op (1) : evt. nu <1, dus een X_{tf} moet weer worden opgehoogd, heeft evt. gevolgen voor (3)-(5).

op (2) : stel s=h(q), dan wordt Y_{qsf} 0, heeft gevolgen voor (2) en dan evt. voor (5).

$\underline{2}$: op (3) : evt. nu $>C_s$, dus een X_{sf} moet worden verlaagd, zie nu weer (1).

op (1) : stel s=h(q), dan Y_{qsf} wordt 1, heeft gevolgen voor (2) en dan evt. voor (5).

op (5) : evt. nu $>CC_{st}$, dan een Y_{qsf} of $X_{t,f(u)}$ verlagen, zie nu $\underline{1}$.

Uit het bovenstaande volgt dat het afronden van een gebroken oplossing tot een toegelaten gehele oplossing ingewikkeld en wellicht niet mogelijk is.

Vuistregels kunnen zijn:

i) Rondt eerst die vaiabelen af die het dichtst bij 0 of 1 liggen.

ii) Rondt eerst die variabelen af waarvoor de beperkingen de meeste speling hebben (hier in de gevallen (1),(3) en (5)).

Met behulp van de 'general assignment eigenschap' van Benders ([BEN83]) en de daarbij gegeven bewijsmethode is gepoogd een bovengrens voor het aantal gebroken variabelen in een oplossing van het gerelaxeerde probleem te vinden: dit leidde echter slechts tot triviale afschattingen, redenen daarvoor zijn dat Benders' probleem een ongerepliceerde allocatie inhoudt (=-teken in (1), i.p.v.$\geq$-teken) en de extra beperking (5).

4.2.2 Responstijden/Filebeschikbaarheid.

Door bepaalde aannamen, bijv. de modellering van een kanaal als een wachtrij, zijn uitdrukkingen voor de gemiddelde responstijd op een query evt. als functie van de variabelen te verkrijgen (met grote waarschijnlijkheid niet-lineaire termen) en op te nemen als beperking. Dit maakt het oplossen van het model een stuk complexer. Voor de hand ligt, zeker als slechts door simulatie schattingen verkregen kunnen worden, oplossingen van (I) op heuristische wijze tot toegelaten oplossingen te herleiden wat betreft de responstijden.

Beperking (5) kan ook gebruikt worden om de gemiddelde responstijden in de hand te houden. In plaats van C_{st} kunnen we $\rho_{st} \cdot C_{st}$ nemen, met ρ_{st} de maximale 'congestie' (bezettingsgraad in wachtrijtermen) die voor kanaal s→t bereikt mag worden. Afhankelijk van de modellering van het kanaalverkeer als wachtrij kan ρ_{st} dusdanig worden gekozen dat de gemiddelde responstijden acceptabel blijven.

Onder dezelfde aannamen als [CHU73] en de daar gegeven filebeschikbaarheids-beperkingen kan worden nagegaan of deze overschreden worden in een oplossing van (I) en kan een oplossing heuristisch worden aangepast.

Algemeen kunnen oplossingen van (I) op heuristische wijze tot wat performance criteria betreft acceptabele oplossingen worden herleid door een verbeterde performance af te wegen tegen de extra kosten.

4.3 Model II : Aggregeren van query- en updateverkeer per file.

Definiëer:

$$\lambda_{sf} := \sum_{q,h(q)=s} \varphi_q \cdot r_q(f) \cdot l_{qf} \quad ,\forall_{s,f} \cdot$$

Dan is λ_{sf} de totale hoeveelheid eenheden query-verkeer vanuit s voor file f per tijdseenheid. Analoog voor het update-verkeer wordt gedefiniëerd:

$$\psi_{sf} := \sum_{\substack{u,h(u)=s \\ \text{en } f(u)=f}} \varphi_u \cdot l'_u \quad ,\forall_{s,f} \cdot$$

Verder Y_{sft} := de proportie van het queryverkeer vanuit s op f die aan de copie van f in t is toegewezen.

We formuleren nu het mixed-integer LP-probleem (II):

minimaliseer :

$$\underset{\text{(opslag)}}{\sum_s \sum_f \sigma_{sf} \cdot X_{sf}} + \underset{\text{(update)}}{\sum_t \sum_f \left(\sum_s \psi_{sf} \cdot \tau_{st} \right) \cdot X_{tf}} + \underset{\text{(query)}}{\sum_s \sum_t \sum_f \lambda_{sf} \cdot \tau_{st} \cdot Y_{sft}}$$

onder voorwaarde van :

(1) $\sum_s X_{sf} \geq 1 \quad ,\forall_f$ {minimaal één copie}

(2) $\sum_t Y_{sft} + X_{sf} = 1 \quad ,\forall_{s,f}$ {query zo mogelijk aan host-site toewijzen}

(3) $\sum_f L_f \cdot X_{sf} \leq C_s \quad ,\forall_s$ {opslagcapaciteit}

(4) $Y_{sft} \leq X_{tf} \quad ,\forall_{s,t,f}$ {alleen toewijzen aan aanwezige copie}

(5) $\sum_f \lambda_{tf} \cdot Y_{tfs} + \sum_f \psi_{sf} \cdot X_{sf} \leq CC_{st} \quad ,\forall_{s,t}[s \neq t]$ { kanaalcapaciteit }

(6) $X_{sf} = 0 \mid 1 \; ; \; Y_{sft} \geq 0 \quad ,\forall_{s,f,t}$

(II) is nu een zelfde soort probleem als (I) en er gelden analoge opmerkingen omtrent de oplossing ervan.

4.4 Het aantal variabelen en restricties.

Definiëer FQ:= het gemiddeld aantal files dat een query raadpleegt. Nu geldt voor het aantal variabelen en restricties voor de problemen (I) en (II) :

	# variabelen		# resticties
	gehele	continu	
(I)	$n \cdot m$	$n \cdot \lvert Q \rvert \cdot FQ$	$m + 2 \cdot \lvert Q \rvert \cdot FQ + n^2$
(II)	$n \cdot m$	$n^2 \cdot m$	$m + n \cdot m + 2 \cdot n^2$

Aannemelijk is dat $|Q| >> n$, dus (I) zal in het algemeen meer variabelen hebben.

4.5 Speciale structuren.

We zullen nu bekijken wat het effect is van het weglaten van één of meer beperkingen, eventueel in combinatie met andere aannamen :

i) Beperking (3) en (5) weg : (de capaciteitsbeperkingen)
De problemen (I) en (II) zijn nu te herleiden tot m onafhankelijke file allocatie problemen (voor oplossing zie [CAS72] en [RAM83]).

ii) Beperking (5) weg en gelijke transportkosten voor elk kanaal : Dan zijn de problemen te formuleren als die van de vorm in [CER82] en [CHA82].

iii) Transportkosten verwaarloosbaar en (5) weg : Dan problemen weer isomorf met [CER82] (opm. in (1) zal gelijkheid optreden).

iv) Het verwaarlozen van opslagkosten leidt tot problemen met onveranderde structuur.

5. CONCLUSIES.

Gebleken is dat het niet eenvoudig is of zelfs niet mogelijk om alle aspecten van het file allocatie probleem bevredigend in één model onder te brengen. Datgene wat men wel of niet modelleert zal sterk afhangen van hoe de prioriteiten in een reële situatie zijn. Hoe sterk legt men de nadruk op het kostenaspect ten kosten van een mindere performance van het systeem? Hierin zal een zekere afweging moeten plaatsvinden. Daarnaast kunnen wat minder gemakkelijke kwantificeerbare zaken als privacy of veiligheid van gegevens een rol spelen bij het distribueren of repliceren van de database.
Verder is nog voorbijgegaan aan evt. concurrency control problemen en het dynamische file allocatie probleem. Dat laatste houdt de vraag in hoe men na verloop van tijd als gevolg van veranderde parameterwaarden de oude fileallocatie efficiënt kan wijzigen in een betere.
Gesteld kan worden dat er weinig of geen ervaring is met het toepassen van de modellen in reële situaties. In de literatuur wordt hiervan weinig melding gemaakt. Toch kunnen de verschillende soorten modellen en benaderingen met de gegeven oplostechnieken in een praktische situatie een goed hulpmiddel zijn om files in een gedistribueerd database systeem zo optimaal mogelijk te kunnen alloceren.

REFERENTIES.

[APE81] P.M.G.Apers,"Query processing and data allocation in distributed database systems",Mathematisch Centrum Amsterdam,Amsterdam, 1981.

[BEN83] J.F.Benders en J.A.E.E.van Nunen,"A property of assignment type mixed integer linear programming problems",Operations Research Letters,vol.2,no.2,1983,p.47- .

[CAS72] R.G.Casey,"Allocation of copies of a file in an information network", AFIPS,SJCC,vol.40,1972,p.617-625.

[CER82] S.Ceri,G.Martella and S.Pelegatti,"Optimal file allocation in a computer network: a solution based on the knapsack problem", Comp.Networks,vol.6,no.5,1982,p.345-357.

[CER83] S.Ceri,S.Navathe and G.Wiederhold,"Distribution design of logical database schemes", IEEE Trans. Softw. Eng.,vol.SE-9,no.4,1983,p. 487-504.

[CHA76] K.M.Chandy and J.E.Hewes,"File allocation in distributed systems" ,in Proc.Int.Symp.on Computer Performance Modelling,Measurement, and Evaluation (Cambridge,Mass.,mrt 1976),p.10-13

[CHA82] S-K.Chang and A-C.Liu,"File allocation in a distributed database" ,Int.J.Comput.and Inf.Sciences,vol.11,no.5,1982,p.325-340.

[CHE80] P.P-S.Chen and J.Akoka,"Optimal design of distributed information systems",IEEE Trans.Computers,vol.C-29,no.10,1980,p.1068-1080.

[CHU69] W.W.Chu,"Optimal file allocation in a multiple computer system", IEEE Trans.Computers,vol.C-18,no.10,1969,p.885-889.

[CHU73] W.W.Chu,"Optimal file allocation in a computer network",in Computer-Communications Systems,N.Abrahamson and F.F.Kuo,Eds., Prentice Hall, Englewood Cliffs,N.J.,1973,p.82-94.

[DOW82] L.W.Dowdy and D.V.Foster,"Comparative models of the file assignment problem", Computing Surveys, vol.14, no.2,1982, p.287-314. (overzicht)

[EFR66] M.A.Efroymson and T.C.Ray,"A branch and bound algorithm for plant location",Oper.Res.,1966,p.361-368.

[ESW74] K.P.Eswaran,"Placements of records in a file and file allocation in a computer network",in Proc.IFIPS Conf.(Stockholm,Sweden,aug. 1974),p.304-307.

[FIS80] M.L.Fisher and D.Hochbaum,"Database location in computer networks",J.ACM,vol.27,okt 1980,p.718-735.

[GRA77] E.Grapa and G.G.Belford,"Some theorems to aid in solving the file allocation problem",Commun.ACM,vol.20,no.11,1977,p.878-882.

[IRA81] K.B.Irani and N.G.Khabbaz,"A combined communication network design and file allocation for distributed databases",in Proc.3rd Int. Conf.Distrib.Comput.Syst.,1981,p.197-209.

[LAN83] L.J.Laning and M.S.Leonard,"File allocation in a distributed computer communication network",IEEE Trans.Computers,vol.C-32,no.3, 1983,p.232-244.

[MAH76] S.Mahmoud and J.S.Riordon,"Optimal allocation of resources in distributed information networks",ACM Trans.Data Base Syst.,vol.1 ,no.1,1976,p.66-78.

[MOR76] H.L.Morgan and K.D.Levin,"Optimal program and data location in computer networks",Commun.ACM,vol.20,no.5,1976,p.315-322.

[RAM79] C.V.Ramamoorthy and B.W.Wah,"The placement of relations in a distributed database",in Proc.1st Int.Conf.Distrib.Comput.Syst., 1979,p.642- 649.

[RAM83] C.V.Ramamoorthy and B.Wah,"The isomorphism of simple file allocation",IEEE Trans.Computers,vol.C-32,no.3,1983,p.221-232.

[WAH84] B.W.Wah,"File placement in distributed computer systems", IEEE Computer,vol.17,no.1,1984,p.23-33.(overzicht)

In this series appeared :

No.	Author(s)	Title
85/01	R.H. Mak	The formal specification and derivation of CMOS-circuits
85/02	W.M.C.J. van Overveld	On arithmetic operations with M-out-of-N-codes
85/03	W.J.M. Lemmens	Use of a computer for evaluation of flow films
85/04	T. Verhoeff H.M.J.L. Schols	Delay insensitive directed trace structures satisfy the foam rubber wrapper postulate
86/01	R. Koymans	Specifying message passing and real-time systems
86/02	G.A. Bussing K.M. van Hee M. Voorhoeve	ELISA, A language for formal specifications of information systems
86/03	Rob Hoogerwoord	Some reflections on the implementation of trace structures
86/04	G.J. Houben J. Paredaens K.M. van Hee	The partition of an information system in several parallel systems
86/05	Jan L.G. Dietz Kees M. van Hee	A framework for the conceptual modeling of discrete dynamic systems
86/06	Tom Verhoeff	Nondeterminism and divergence created by concealment in CSP
86/07	R. Gerth L. Shira	On proving communication closedness of distributed layers

86/08	R. Koymans R.K. Shyamasundar W.P. de Roever R. Gerth S. Arum Kumar	Compositional semantics for real-time distributed computing (Inf. & Control 1987)
86/09	C. Huizing R. Gerth W.P. de Roever	Full abstraction of a real-time denotational semantics for an OCCAM-like language
86/10	J. Hooman	A compositional proof theory for real-time distributed message passing
86/11	W.P. de Roever	Questions to Robin Milner - A responders commentary (IFIP86)
86/12	A. Boucher R. Gerth	A timed failures model for extended communicating processes
86/13	R. Gerth W.P. de Roever	Proving monitors revisited: a first step towards verifying object oriented systems (Fund. Informatica IX-4)
86/14	R. Koymans	Specifying passing systems requires extending temporal logic
87/01	R. Gerth	On the existence of a sound and complete axiomatizations of the monitor concept
87/02	Simon J. Klaver Chris F.M. Verberne	Federatieve Databases
87/03	G.J. Houben J. Paredaens	A formal approach to distributed information systems
87/04	T. Verhoeff	Delay-insensitive codes - An overview
87/05	R. Kuiper	Enforcing non-determinism via linear time temporal logic specification

87/06	R. Koymans	Temporele logica specificatie van message passing en real-time systemen (in Dutch)
87/07	R. Koymans	Specifying message passing and real-time systems with real-time temporal logic
87/08	H.M.J.L. Schols	The maximum number of states after projection
87/09	J. Kalisvaart L.R.A. Kessener W.J.M. Lemmens M.L.P van Lierop F.J. Peters H.M.M. van de Wetering	Language extensions to study structures for raster graphics
87/10	T. Verhoeff	Three families of maximally nondeterministic automata
87/11	P. Lemmens	Eldorado ins and outs. Specifications of a data base management toolkit according to the functional model
87/12	K.M. van Hee A. Lapinski	OR and AI approaches to decision support systems
87/13	J. van der Woude	Playing with patterns, searching for strings
87/14	J. Hooman	A compositional proof system for an occam-like real-time language
87/15	G. Huizing R. Gerth W.P. de Roever	A compositional semantics for statecharts
87/16	H.M.M. ten Eikelder J.C.F. Wilmont	Normal forms for a class of formulas
87/17	K.M. van Hee G.J. Houben J.L.G. Dietz	Modelling of discrete dynamic systems framework and examples

87/18	C.W.A.M. van Overveld	An integer algorithm for rendering curved surfaces
87/19	A.J. Seebregts	Optimalisering van file allocatie in gedistribueerde database systemen