On Building Integrated and Distributed Database Systems

Data Allocation

Robert Wrembel
Poznań University of Technology
Institute of Computing Science
Poznań, Poland
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

Data allocation problem

- Data allocation
  - storing data (eg. tables, fragments of tables, replicas) in nodes of a distributed DB
- Uses
  - fragmentation
  - replication
- Considers
  - workload of nodes
    - data reads ⇨ data should be "close" to their "consumers"
    - data modifications (I, U, D) ⇨ need for synchronizing replicas

Data allocation problem

- Originates from the problem of allocating files in a computer network
- NP-complete problem

  n fragments, m nodes: $(2^m - 1)^n$

- Benefits from allocation depends on
  - the "quality" of allocation
  - capabilities/functionalities of a query optimizer

Into which nodes to replicate?

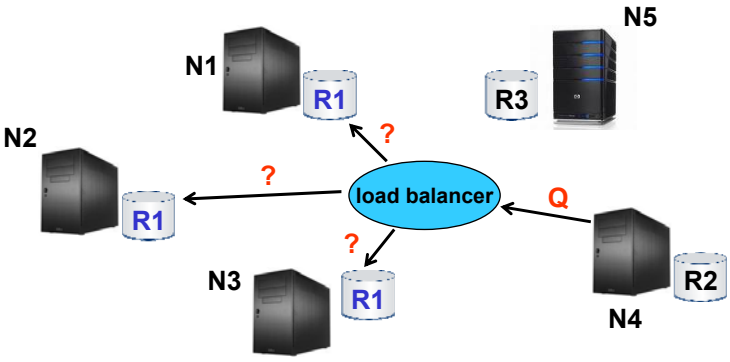
- Allocation algorithm has to take into consideration
  - characteristics of queries in nodes
  - data transmission (replica refreshing) costs between data
  - storage costs
  - computing power of nodes
- The problem is NP-complete
  - no exact algorithms
  - heuristics are applied

Into which nodes to replicate?

- Multiple heuristics
  - M. T. Özsu, P. Valduriez, Distributed and Parallell Database Systems, ACM Computing Surveys, vol. 28, no. 1, 1996, pp.125-128
  - Y-F. Huang, J-H. Chen, Fragment Allocation in Distributed Database Design, July 2000, pp. 491-506
  - A. Brunstrom, S. T. Leutenegger, R. Simha, Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database With Changing Workloads, no. TR-95-2, 1995, pp. 1-15
  - P. M. G. Apers, Data Allocation in Distributed Database Systems, ACM Transactions on Database Systems, vol. 13, no. 3, September 1988, pp. 263-304
- Static and dynamic algorithms

How to balance load?

- Replica R1 is stored in nodes N1, N2, and N3
- Query Q issued at node N4 is going to access R1
- Select the node that answers Q in the shortest time
- Select the node that a workoad is evenly distributed between nodes



How to balance load?

- Load balancer should know
  - current load for each node
  - metadata for query optimization at each node (data structures, histograms, data sizes and allocation parameters, ...)
  - communication bandwidth with each node
  - processing speed of each node
  - query optimizer capabilities of each node
  - discs transfer speed at each node
  - ...
  - heuristics
- Possible solution
  - competing nodes ⇨ execute Q at all nodes that store R1 and stop operations (after a while) on the nodes that respond with the largest delay

Best Fit algorithm

- Intuitive approach
- Fragment Fi is stored in this node Sj, where the number of reads and modifications of Fi is the greatest
- Constraint: no replication ⇨ fragment is allocated in only ONE node

- Example
  - processing characteristic
  - nodes S1 i S4 access F1 i F2 with frequency 1
    - F1: 3 reads and 1 modification
    - F2: 2 reads
  - ...

Node	Freq.	No accesses
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3w+1w) do **F3**: (2r)

Best Fit - example

➲ **Processing charact.**

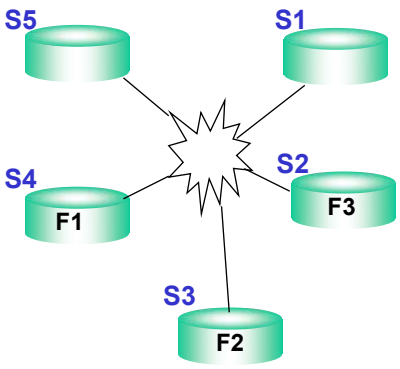
Node	Freq.	No accesses
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3w+1w) do **F3**: (2r)

Fragment	Node	Trans. T1	Trans. T2	Trans. T3	Ref. sum
F1	S1	1*(3r+1w)	0	0	4
	S2	0	2*2r	0	4
	S3	0	0	0	0
	S4	1*(3r+1w)	2*2r	0	8
	S5	0	0	0	0

Fragment	Node	Trans. T1	Trans. T2	Trans. T3	Ref. sum
F2	S1	1*2r	0	0	2
	S2	0	0	0	0
	S3	0	0	3*(3r+1w)	12
	S4	1*2r	0	0	2
	S5	0	0	3*(3r+1w)	12

Fragment	Node	Trans. T1	Trans. T2	Trans. T3	Ref. sum
F3	S1	0	0	0	0
	S2	0	2*(3r+1w)	0	8
	S3	0	0	3*2r	6
	S4	0	2*(3r+1w)	0	8
	S5	0	0	3*2r	6

Best Fit - example



Best Fit

- Computationally simple and intuitive
- Does not use replication
- Small "accuracy"
  - only access frequencies are taken into consideration

Profit optimization algorithm

- Algorytm selects all nodes, where it is profitable to place a fragment ⇨ supports replication
- Fragment Fi is placed in these nodes where profit from placing Fi there is greater than storage and modification costs
- Fragment Fi is placed in node Sj, where read cost is greater than write cost of Fi from any node in the system

Profit optimization algorithm

- **Cost** of maintaining additional copy of fragment **Fi** in node **Sj** is computed as:
  - total time of all local modifications of **Fi** in **Sj** **+**
  - total time of all remote modifications of **Fi** from remote nodes (other than **Sj**)
- **Profit** of maintaining additional copy of fragment **Fi** in node **Sj** is computed as:
  - remote query execution time (without replication) **-** local query execution time ***** frequency of querying **Fi** from **Sj**

Example

- **Processing characteristics**

Fragment	Size	AVG **local** query time (modification) [ms]	AVG **remote** query time (modification) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Node	Freq.	No accesses
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3w+1w) do **F3**: (2r)

- **Modification costs for allocation schemas of fragment F1**
  - **F1 is allocated subsequently in S1, S2, ..., S5**

Fragment	Węzeł	Modyfikacje lokalne i zdalne	częstość * liczba mod. * czas	Koszt
F1	S1	zdalna z S4 (lokalna w S1)	1*1*600 (zdalna) + 1*1*150 (lokalna)	750
	S2	zdalna z S1 zdalna z S4	1*1*600 (zdalna) + 1*1*600 (zdalna)	1200
	S3	zdalna z S1 zdalna z S4	1*1*600 (zdalna) + 1*1*600 (zdalna)	1200
	S4	zdalna z S1 (lokalna w S4)	1*1*600 (zdalna) + 1*1*150 (lokalna)	750
	S5	zdalna z S1 zdalna z S4	1*1*600 (zdalna) + 1*1*600 (zdalna)	1200

Example

➲ **Processing characteristics**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

➲ **Modification costs for allocation schemas of fragment F2**

- **F2 is allocated subsequently in S1, S2, ..., S5**

Fragment	Węzeł	Modyfikacje lokalne i zdalne	częstość * liczba mod. * czas	Koszt
F2	S1	zdalna z S3 zdalna z S5	3*1*700 (zdalna) + 3*1*700 (zdalna)	4200
	S2	zdalna z S3 zdalna z S5	3*1*700 (zdalna) + 3*1*700 (zdalna)	4200
	S3	zdalna z S5 lokalna w S3	3*1*700 (zdalna) + 3*1*200 (zdalna)	2700
	S4	zdalna z S3 zdalna z S5	3*1*700 (zdalna) + 3*1*700 (zdalna)	4200
	S5	zdalna z S3 lokalna w S5	3*1*700 (zdalna) + 3*1*200 (zdalna)	2700

Example

➲ **Processing characteristics**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

➲ **Modification costs for allocation schemas of fragment F3**

- **F3 is allocated subsequently in S1, S2, ..., S5**

Fragment	Węzeł	Modyfikacje lokalne i zdalne	częstość * liczba mod. * czas	Koszt
F3	S1	zdalna z S2 zdalna z S4	2*1*1100 (zdalna) + 2*1*1100 (zdalna)	4400
	S2	zdalna z S4 lokalna w S2	2*1*1100 (zdalna) + 2*1*250 (lokalna)	2700
	S3	zdalna z S2 zdalna z S4	2*1*1100 (zdalna) + 2*1*1100 (zdalna)	4400
	S4	zdalna z S2 lokalna w S4	2*1*1100 (zdalna) + 2*1*250 (lokalna)	2700
	S5	zdalna z S2 zdalna z S4	2*1*1100 (zdalna) + 2*1*1100 (zdalna)	4400

Example

- **Processing characteristics**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

- **Profit for the allocation schema of fragment F1**

Fragment	Węzeł	Zapytanie	częstość * liczba mod. * czas	Zysk
F1	S1	z S1	1*3*(500-100)	1200
	S2	z S2	2*2*(500-100)	1600
	S3	brak	0	0
	S4	z S4 z S4	1*3*(500-100) + 2*2*(500-100)	2800
	S5	brak	0	0

Example

- **Processing characteristics**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

- **Profit for the allocation schema of fragment F2**

Fragment	Węzeł	Zapytanie	częstość * liczba mod. * czas	Zysk
F2	S1	z S1	1*2*(650-150)	1000
	S2	brak	0	0
	S3	z S3	3*3*(650-150)	4500
	S4	z S4	1*2*(650-150)	1000
	S5	z S5	3*3*(650-150)	4500

Example

- **Processing characteristics**

Fragment	Rozmiar	Śr. **lokalny** czas zapytania (aktualizacji) [ms]	Śr. **zdalny** czas zapytania (aktualizacji) [ms]
F1	300 KB	100 (150)	500 (600)
F2	500 KB	150 (200)	650 (700)
F3	1 MB	200 (250)	1000 (1100)

Węzeł	Częstość	L. dostępów
S1,S4	1	do **F1**: (3r+1w) do **F2**: (2r)
S2,S4	2	do **F1**: (2r) do **F3**: (3r+1w)
S3,S5	3	do **F2**: (3r+1w) do **F3**: (2r)

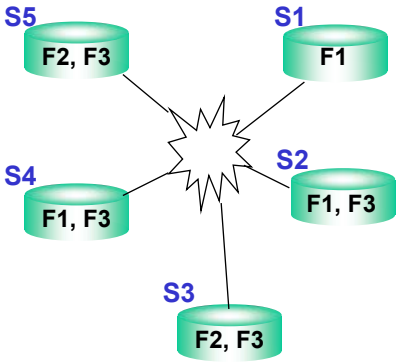
- **Profit for the allocation schema of fragment F3**

Fragment	Węzeł	Zapytanie	częstość * liczba mod. * czas	Zysk
F3	S1	brak	0	0
	S2	z S2	2*3*(1000-200)	4800
	S3	z S3	3*2*(1000-200)	4800
	S4	z S4	2*3*(1000-200)	4800
	S5	z S5	3*2*(1000-200)	4800

Example

- **Final allocation schema**
  - **fragment Fi is allocated in all the nodes where profit>cost**

Fragment	Węzeł	Koszt	Zysk
F1	**S1**	750	**1200**
	S2	1200	**1600**
	S3	1200	0
	S4	**750**	**2800**
	S5	1200	0
F2	S1	4200	1000
	S2	4200	0
	S3	**2700**	**4500**
	S4	4200	1000
	S5	**2700**	**4500**
F3	S1	4400	0
	S2	**2700**	**4800**
	S3	**4400**	**4800**
	S4	**2700**	**4800**
	S5	**4400**	**4800**



Problem formulation

- The following information is given
  - characteristics and frequencies of queries
  - characteristics and frequencies of DML operations
  - network nodes
  - throughput of media connecting nodes
  - processing power of nodes
- The following is being searched
  - allocation schema of fragments in nodes (redundancies are allowed) so that a given cost function is minimal / maximal

Required data

- Fragments $F = \{F_1, F_2, ..., F_j\}$
  - the size of every fragment (F_j) must be known ⇨ network communication costs
- Transactions $T = \{T_1, T_2, ..., T_i\}$
  - type (read, write)
  - frequencies of executions
  - subsets of accessed data
- Nodes $S = \{S_1, S_2, ..., S_k\}$
  - discs capacities
  - I / O characteristics
  - CPU, ...
- Network
  - throughput

Info about transactions

- RM (Retrieval Matrix)

	F1	F2	F3	F4	F5
T1	2	3	0	0	0
T2	2	0	0	1	0
T3	0	0	3	0	0
T4	3	0	2	0	0

- UM (Update Matrix)

	F1	F2	F3	F4	F5
T1	0	0	0	1	2
T2	0	3	0	0	0
T3	2	1	0	1	0
T4	0	0	0	0	3

Info about transactions

- Not all rows must be updated or read (Selectivity Matrix)

SEL:(%)

	F1	F2	F3	F4	F5
T1	0.1	0.1	0	0.3	0.2
T2	0.1	0.3	0	1	0
T3	2	5	0.1	0.5	0
T4	0.5	0	10	0	4

- Access Frequency Matrix

FRQ:

	S1	S2	S3	S4
T1	0	2	3	1
T2	0	3	0	0
T3	2	0	1	0
T4	0	0	4	0

Info about network

- Communication Cost Matrix - represents throughput between nodes
- Simplification: CCM is symmetrical

CCM

	S1	S2	S3	S4
S1	0	0.32	0.48	0.16
S2	0.32	0	0.64	0.32
S3	0.48	0.64	0	0.64
S4	0.16	0.32	0.64	0

Cost

- A cost may include
  - the number of I/O operations (data volume)
  - processor time
  - network communication costs (data transfer)
  - the size of a data buffer (cache)
  - ...

Cost

- Cost minimization may include
  - cost of storing fragment Fj in node Sk
  - cost of executing queries on Fj in node Sk
  - cost of modifying Fj in all the nodes where Fj is stored
  - cost of network communication
- Optimization criteria
  - minimization of response time
  - maximization of trhougput of each node

Static algorithms

- Executed during system's idle time
- Executed periodically
- Computationally complex
  - $O(jk^2i)$
    - j - number of table fragments
    - k - number of nodes storing data
    - i - number of transactions

Example algorithm

- **Optimization goal**
  - **allocate fragments in nodes so that:**

$$\min(CC_{transfer} + CC_{transakcji})$$

$CC_{transfer}$ – **data transfer cost**
$CC_{transakcji}$ – **transaction execution cost**

- **Input data**
  - **RM(Ti, Fj) – Retrieval matrix**
  - **UM(Ti, Fj) – Update matrix**
  - **SEL(Ti, Fj) – Selectivity matrix**
  - **FRQ(Ti, Sk) – Frequency matrix**
  - **CCM(Sk, Sm) – Communication cost matrix**
- **Output data**
  - **FAT(Fj, Sk) – Fragment Allocation Table**

Example algorithm

- **Initiate the FAT (Fragment Allocation Table) matrix**
- **Algorithm - step 1**
  - **for each transaction Ti, fragment Fj, node Sk do:**
    - IF
      - the number of accesses to Fj by Ti in Sk and
      - execution frequency of Ti in Sk > 0
    - THEN leave the copy of fragment Fj in Sk

Example algorithm

Step 1
For Ti in T, Fj in F, Sk in S do
if **(**RM**(Ti,Fj) *** FREQ**(Ti,Sk) > 0)**
FAT**(Fj,Sk) := 1**

RM

	F1	F2	F3	F4	F5
T1	2	3	0	0	0
T2	2	0	0	1	0
T3	0	0	3	0	0
T4	3	0	2	0	0

FREQ:

	S1	S2	S3	S4
T1	0	2	3	1
T2	0	3	0	0
T3	2	0	1	0
T4	0	0	4	0

FAT - krok 1

	F1	F2	F3	F4	F5
S1	0	0	0	0	0
S2	1	1	0	1	0
S3	1	1	1	0	0
S4	1	1	0	0	0

Example algorithm

- **Remove copies of allocated fragments in order to reduce refreshing costs**
  - **remove copies for which profit < cost**
- **Algorithm - step 2**
  - **for each Fj allocated in more than one node, repeat the insertion procedure until for each fragment profit > cost or there exists only one copy of Fj in DDBS**
  - **if profit < cost in all nodes then leave Fj in this node where the value profit-cost is the greatest**

Step 2
For Fj in F do
While (NumFragCopy**(Fj) > 1)**
begin
Let Sk be the site with FAT**(Fj,Sk)=1**
and a minimum value of (Benefit**(Fj,Sk)-**Cost**(Fj,Sk));**
if **((**Benefit**(Fj,Sk) -** Cost**(Fj,Sk)) < 0)**
FAT**(Fj,Sk) = 0**
end;

Example algorithm

Cost(Fj,Sk)

	F1	...	F5
S1	0	...	0
S2	6	...	0
S3	4	...	0
S4	3	...	0

Benefit(Fj,Sk)

	F1	...	F5
S1	0	...	0
S2	4	...	0
S3	7	...	0
S4	9	...	0

FAT from step 1

	F1	F2	F3	F4	F5
S1	0	0	0	0	0
S2	1	1	0	1	0
S3	1	1	1	0	0
S4	1	1	0	0	0

FAT - step 2

	F1	...	F5
S1	0	...	0
S2	0	...	0
S3	1	...	0
S4	1	...	0

Example algorithm

- **Check if all fragments have been allocated**
- **Algorithm - step 3**
  - **if there exists a non allocated fragment that is being accessed, then allocate the fragment in these node where its accessing cost is the lowest**

```
For Fj in F do
    if (NumFragCopy(Fj) = 0 and UM(Ti,Fj) > 0)
      begin
        Sk = MinOperCost(Fj);
        FAT(Fj,Sk) = 1;
      end;
```

Example algorithm

UM

	F1	F2	F3	F4	F5
T1	0	0	0	1	2
T2	0	3	0	0	0
T3	2	1	0	1	0
T4	0	0	0	0	3

FREQ:

	S1	S2	S3	S4
T1	0	2	3	1
T2	0	3	0	0
T3	2	0	1	0
T4	0	0	4	0
sum:	0	2	7	1

FAT - step 2

	F1	...	F5
S1	0	...	0
S2	0	...	0
S3	1	...	0
S4	1	...	0

FAT - step 3

	F1	...	F5
S1	0	...	0
S2	1	...	0
S3	0	...	1
S4	0	...	0

Example algorithm - alternative version

- Steps 1 and 3 are the same as in the previous version of the algorithm
- Step 2 - removing fragments
  - order all fragments by their decreasing weights
  - the weight is computed based on the number of updates made by transactions initiated in remote nodes
  - a copy is removed from a node when its weight is greater

Dynamic algorithms

- Applicable when the workload arriving to nodes and access characteristics to data change in time
- Static algorithms may deteriorate DDBS performance
- Dynamic algorithms support fragment relocation
- Dynamic algorithms
  - Simple Counter
  - Load Sensitive Counter

Simple Counter

- One dedicated node maintains access counters for each fragment [$Node_i$, $Fragment_i$]
- An access counter counts the number of times a fragment is being accessed
- A system process periodically checks counters for each fragment
- Fragment F_i is relocated to the node with the greatest value of a counter

Simple Counter

- The system process
  - stores also statistics on: throughput, average response time, the number of transactions accessing fragments
- The frequency of checking counters is crucial
  - must be high enough to be able to follow changes in a workload
  - must not be too high in order to not continuously relocate data

Simple Counter

- Advantages:
  - offers sufficient allocation scheme when
    - a workload is relatively low
    - the load of all nodes is similar
    - workload changes stabilize in time
- Disadvantages:
  - if all transactions come from the same node then all fragments may be relocated to this node ⇨ node overloading

Load Sensitive Counter

- Monitors system's load and the frequency of accessing fragments
- Fragment relocation is executed by the Simple Counter Algorithm
- Fragments are relocated only if a node load remains below a given threshold
- Algorithm parameters
  - maximum % of data stored in a node
  - maximum node load

Dynamic algorithms

- Features
  - not efficient when the frequency of counter checking is too high
  - not efficient when the workload changes quickly
  - whe worst efficiency is when the system relocates fragments trying to follow the changing workload (too much fragment relocations)

Literature

- M. T. Özsu, P. Valduriez, Distributed and Parallell Database Systems, ACM Computing Surveys, vol. 28, no. 1, 1996, pp.125-128
- Y-F. Huang, J-H. Chen, Fragment Allocation in Distributed Database Design, July 2000, pp. 491-506
- A. Brunstrom, S. T. Leutenegger, R. Simha, Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database With Changing Workloads, no. TR-95-2, 1995, pp. 1-15
- P. M. G. Apers, Data Allocation in Distributed Database Systems, ACM Transactions on Database Systems, vol. 13, no. 3, September 1988, pp. 263-304