Agoric paradigm for wide area distributed database management

Mariposa Principles

“Centralized database managers will be an antique curiosity by the year 2000” Stonebraker 1988

- Scalability to a large number of cooperating sites
- Data Mobility
- No global synchronization
- Total local autonomy
- Easily configurable policies

Requirements for DBMS in non-uniform, multi-administrator WAN env:

1: several hundred sites – highly scalable
2: Items have no notion of home, only of current owner. Fragments are collections of records that belong to a common DBMS class
3: sites can create/delete objects, two sites can agree to move an object without notifying the world
4: so sites can refuse to process queries for other sites
5: want a high level of flexibility in policy design (heterogeneous) storage manager is rule-driven and based on the reception of events

others:
- no difference between dist storage and deep storage ie so moving from backup to main memory is same as moving between site -> therefore one logical site per storage device
- support for data to query or query to data
- support for permanent and temporary storage of copies
General Overview (1)

• Microeconomic paradigm adopted – each site is seeks to maximise their profit per unit operating time

• Mariposa sites have accounts with a network bank: clients, servers and brokers

• Clients submit queries with a budget attached

• Goal: try to process the query within the budget $B(t)$ by subcontracting to various sites

• Queries are administered by a broker

paradigm adopted for storage and query optimisation
DBMS issues reformulated in economic terms: multiple copies of objects and name services
No need for a centralised admin service, this is provided by the “invisible hand” of market forces
This also means it dynamically adapts to resource contention
Clients, servers and brokers can be at the same site or may be distributed across many sites
broker expends the budget in a way that balances resource usage with query response time
Storage manager decides to buy or sell fragments of data
General Overview (2)

- Broker’s bid manager tries to find processing sites (ie servers) which fit the client’s budget
- Fragments are the basic unit of data – may be split or coalesced
- Storage Mangers decide which fragments to buy and sell and which queries to bid on
- Administrators may alter behaviour by changing rules at their site
Architecture

- Middleware: several query preparation modules, query broker, coordination module
- Local to each site: local execution engine, bid manager, storage manager
- Query parser requests metadata from the servers
- Broker: two step process for query execution plan
  - Broker's bid manager starts a bidding process for the strides of the query

name servers provide name and type of attributes in the class and relevant stats. parser's choice of name server is based on qos and budgetary constraints

Broker prepares query execution plan: single site optimiser and then fragmented plan using metadata

Breaks things up into parallelisable STRIDES

Locally defined rules may affect how the sub queries are broken up

Storage managers, bidders, brokers coded in a rule language called Rush
mariposa has a specific inter site protocol which all sites use to communicate.
Also have the canonical representation of data which can be used or must be translated from/to local language.
Bidding Process

- The budget $B(t)$ is a non-increasing function of time

- Resolution of each sub-query $Q_i$ by either an expensive bid protocol or purchase order protocol

Expensive Bid Protocol
- Send RFB, receives triples $(C_i, D_i, E_i)$
- Broker notifies winning bidder – high message overhead

Purchase Order Protocol
- Broker sends subqueries $Q_i$ directly to the sites most likely to win a bidding process if there was one

-constant functions indicate the willingness to pay the same no matter when the query delay

-most queries use the simpler purchase order protocol

-bidder will resolve the sub-query $Q_i$ for a cost $C_i$ within a delay $D_i$ and bid is valid till an expiration time $E_i$

-If there is no bid for some subquery or no collection meets client price and perf broker must get other bids, do it himself or notify client that its impossible
Bid Acceptance

• In acceptance only collections of bids for the subqueries in each stride are considered
• Winning bid must have aggregate cost C and delay D s.t. 
  \[ C = B(D) \]
• To compare collections of bids we have
  difference = B(D)-C

• Greedy heuristic alg for determining the winner of bids. Starts with the collection of smallest delay
• Bidders are found from advertisements – brokers should remember sites which have previously been successful

-All sub queries in each stride are processed in parallel – next stride cannot begin until the previous has finished
- The aggregate delay d is not the sum of delays Di because of parallelism within each stride
  - the estimated delay to process the collection of subqueries in the stride is equal to the highest bid time in the collection
  - difference may be negative if cost is over
- Never do exhaustive search through bids -> explosive
- Greedy Alg is one in which a series of solutions are proposed with increasing delay values for each processing steps. The cost gradient is the cost decr that would result for the proc step by replacing the collection in the solution by the one being considered, divided by the time increase that would come from the substitution. Start with one of smallest delay
Setting Bid Price

- How to set bid price for subqueries?
- Base billing rate for CPU and I/O resources for each site, set by admin
- Improvement 1: to have rate on a per frag basis – hastens sale of low value fragments, bias bids towards desired frags
- Improvement 2: bids based on current load average – provides a crude form of load balancing among the sites
- Improvement 3: Hotlists of frags wanted by site
- The Network bidder for bandwidth queries and network resource reservation, not fully developed in the paper
Storage Management

• Storage filled with fragments or copies of fragments

• Site keeps size and revenue history for each fragment
  History required by potential buyers

• Offer price = value of frag – value of evicted frags + price
  gained from evicted frags

• Market forces to decide size of fragments
  Also available: Expected feasible site bid = B(ED) / Num_c

-Calculating the expected value of frags
-Cpu and i/o info is normalised and stored in site independent units
-revenue history: number of records, query, time-since-last-query, revenue,
delay, resources used
-May buy on demand or prefetch.

too many => overhead increases and response time will suffer
too few frags in the class => parallel exe hindered
consider execution where query accesses a single class, broker fetches
NUMc
Assume frags are the same size, can compute the expected delay (ED) if all
frags run in parel
Budget function then tells ammount available for entrire query under this
delay
Repeat calc and find NUMc which maximises expected revenue
mariposa has a specific inter site protocol which all sites use to communicate
Also have the canonical representation of data which can be used or must be translated from/to local language
Storage Management

- too few frags in the class => parallel exe hindered
- too many => overhead increases and response time will suffer
Name Service

• Four structures used in object naming
  – Internal names: location dependant
  – Full names : uniquely identify an object
  – Common names : shortcuts partially specified
  – Name Contexts : set of affiliated names

• The name service process uses bidding protocol. Translation from common names to full names via client supplied name contexts
• Name resolution is not infallible, in case of ambiguity a “best match” is used
• Name services win bids based on cost and quality of service (freshness of metadata) considerations

1. determines the physical locations
2. full names can be resolved by any object regardless of location
3. translation of common names is performed by functions written in the mariposa rule ext lang, stored in the system catalogs
4. names within a context are expected to same some features. might be used in a directory or a complex object (class definition). Programmers may name global contexts for everyone to use
Summary & Conclusions

- Traditional approaches to distributed database management unsuitable for asynchronous heterogeneous systems with mobile data
- Microeconomic paradigm for handling query and storage optimisation
- Does not seek to enforce globally optimal solutions
- Bidding process shown to not introduce much overhead
- Approach appears to have been commercialised and applied to database integration

Bidding and storage manager quite basic at the time of writing
Network bidder had not been implemented at all
Only 3 databases in the test case!!
Fin
What happened since then?

- Broadband: network speeds ~ I/O speeds get data from neighbours cache rather than your own
- Mobile Computing Environments: type and size of storage may be limited, bandwidth availability vs Communication latency
- Internet: heterogeneity -> lack of schema, sites may not be capable of processing all queries

- Web/Database integration => Java/XML/Active Server Page
- All data => relational => XML
  Xpath, Xquery, XML Schema

plus usual mobile problems: freq disconnects, low bandwidth, noise, echo
standardisation: .NET and J2EE
COM and Corba were in infancy in 96