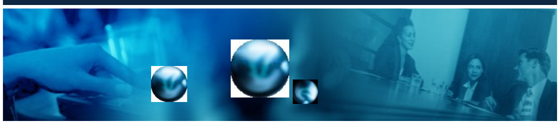
Mariposa:
A wide-area distributed database



Slides originally by Shahed Alam
Edited by Tim Zhu, Feb 02, 2009

Outline

Outline

Motivation

- Build a wide-area Distributed database system
- Single program performing global query optimization using cost-based optimizer will not work well:
  - access constraints
  - charging algorithms
  - site-specific data type extensions
  - scale a large number of possible processing sites
- Traditional distributed DBMS not appropriate in a modern WAN environment
- A new architecture is required

Outline

(wrong) Assumptions in Distributed DBMS

- Static data allocation
  - No handling of changing access patterns
  - Manual transfer of data from site to site
- Single administrator
  - Site selection done by optimizer
  - But what if site belongs to another? Chance of being refused (when site overloaded or indisposed)?
- Uniformity
  - Different hardware, network connections, hard disk space
- Assumptions hold for LAN but not multi-admin WAN

Assumptions for WAN based DDBMS

- Scalability
  - No limit of its ability to scale more sites
- Data mobility
  - Change “home” of object. Available during movement
- No global synchronization
  - Schema changes should not cause sync
- Total local autonomy
  - Sites control own local resources. What objects to store, what queries to run
- Easily configurable policies
  - Easily change individual rules of sites by local administrators

7

Discussion Questions

- -- Suppose you need a huge database for an application. What conditions make you to setup a distributed database? What conditions make you to setup a non-distributed database? Can you think of some applications for the two approaches?
- -- What applications can you imagine using Mariposa for? What about systems using the previous assumptions for distributed system?

Outline

1. Motivation
2. Assumptions for DDBMS
3. Economics in Mariposa

9

Application of economics to Mariposa

- All Mariposa Clients and servers have accounts with a network bank
- User allocates budget to each query
  - Goal: try to process the query within the budget B(t) by subcontracting to various sites
- Query administered by broker which obtains bids
- Fragments (objects) are the units of storage that are bought and sold (may be split or coalesced)
- Servers buy objects, advertise its services, bids on queries, leaves by selling objects
  - Goal: optimize revenue

10

… more economics

- Objects have “current owner” which changes as they are moved
- Object replication based on payment for frequency of updates among copy holders
  - Name servers use the same policy for metadata
- Micro-economic paradigm adopted – each site is seeks to maximize their profit per unit operating time
- Each site has a bidder and storage manager
  - Which objects to buy/sell, which queries to execute
- Administrators may alter behaviour by changing rules at their site

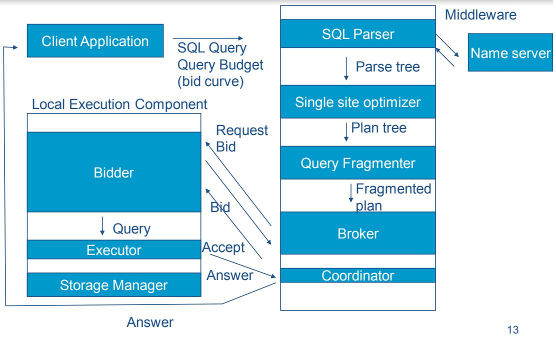
11

Outline

1. Motivation
2. Assumptions for DDBMS
3. Economics in Mariposa
4. Mariposa architecture

12

Mariposa architecture



A few more details…

- Rush
  - Low level, very efficient embedded scripting rule language
  - Every Mariposa entity has a rush interpreter
  - Storage manager, bidder, broker coded in rush

Outline

1. Motivation
2. Assumptions for DDBMS
3. Economics in Mariposa
4. Mariposa architecture
5. Bidding process

Bidding process

- Each query has a budget B(t)
  - B(t) is a non-increasing function of time
- Each query is fragmented into sub-queries
- Strides
  - Multiple fragmented sub-queries that can be executed in parallel
- Broker solves sub-queries using
  - Expensive bid protocol
  - Purchase order protocol

Expensive Bid protocol

- 2 phases
- 1. Request for bids
  - Send portion of query plan being bid
  - Bidder sends back a triple (C,D,E)
    - C= Cost
    - D= Delay (time to process query)
    - E= Expiration date of offer
- 2. notify the winning bidder
- This process used only for complex queries as it is expensive (overhead: too many messages).
- Use Purchase order protocol for simple queries

Purchase order protocol

- Send sub-query to bidder with highest likelihood of winning anyway
  - Keep track of query-history
- Site processes request and sends a “bill”
- Con: Probable budget deficit

Bid Acceptance

- In acceptance only collections of bids for the sub-queries in each stride are considered
- Winning bid must have aggregate cost C and delay D, C = B(D)
- To compare collections of bids we have difference = B(D)-C
- Greedy heuristic algorithm for determining the winner of bids. Starts with the collection of smallest delay

Finding bidders

- Finding bidders
  - Servers post “advertisements” with name servers.
  - Name servers store “ad tables”
    - Advertisements in form of “yellow pages”
    - Example: date of advertisement, sale price, coupons
  - Brokers examine ad tables to locate bidders
  - Brokers remember sites that bid successfully

Setting the bid price

- Bidder sends reply in form (Cost, Delay, Expiration) to broker
  - Cost
    - CPU, I/O (naive), Network resource
    - Optimization: Billing rate per fragment, Adjust cost based on current load, bid on hot list items even if server does not have data
  - Delay
    - Time to process under zero load or current load + safety factor
  - Expiration
    - Set arbitrarily
- Improvements:
  - have rate on a per fragment basis
  - bids based on current load average
  - hot lists of fragments wanted by site

Discussion Question

- Does this bidding process seem like a good model to you? Can you think of other applications for which it might be relevant?

Outline

1. Motivation
2. Assumptions for DDBMS
3. Economics in Mariposa
4. Mariposa architecture
5. Bidding process
6. Storage and Name resolution

Storage Management

- Manages fragments to maximize profits in local execution component
- Buying and selling fragments
  - Maintains history of each fragments revenue
  - Contact current owner for fragment revenue before buying (remember : maximize profit)
  - Performs bidding process to sell fragments that it does not want by sending revenue history to bidders
- Splitting or coalescing fragments
  - Break fragments that have high revenues, to lower copies (to redirect traffic to oneself)
  - Coalesce copies if it takes more processing than is required

Naming and Name service

- Unlike traditional centralized name servers, Mariposa has a DECENTRALIZED name registration system
- Names are unordered sets of attributes
- Each object has four structures for naming
  - Internal names: location dependant
  - Full names : uniquely identify an object
  - Common names : shortcuts partially specified
  - Name Contexts : set of affiliated names

Name resolution and discovery

- Every client-server has local name cache to resolve object names
- Broker queries name-server if match not found
- There exists multiple name-servers
- Broker choose name-server based on quality-of-service (staleness of metadata) required

Discussion Questions

- How does Mariposa compare to today's P2P's system? How is it the same? How is it different?

Outline

1. Motivation
2. Assumptions for DDBMS
3. Economics in Mariposa
4. Mariposa architecture
5. Bidding process
6. Storage and Name resolution
7. Experiment and Conclusion

Experimental Evaluation

- Environment
  - 3 relations in 3 sites, 11MB data
- Test Purchase order Vs Expensive Bid in LAN vs WAN environment
  - Result
  - Broker: 4.52 (s) for PO Vs 14.08 (s) for EB
- Test Expensive Bid to show how data is moved to a closer site for repeated-query
  - Result: all 3 tables move to site that starts the query

Conclusion

- Traditional approaches to distributed database management unsuitable for asynchronous heterogeneous systems with mobile data
- Micro-economic paradigm for handling query and storage optimization
- This model can reduce the scheduling complexity of distributed interactions
- Does not seek to enforce globally optimal solutions
- Bidding process not unduly expensive

Epilogue

- Where is Mariposa now?
  - Mariposa -> Cohera -> PeopleSoft -> Oracle