Wide Area Query Systems … The Hydra of Databases

Stonebraker et al. 96
Gribble et al. 02

Zachary G. Ives
University of Pennsylvania

January 21, 2003
CIS 650 – Data Sharing and the Web

The Vision

- A World Wide Web of autonomous, heterogeneous data sources, each sharing data (tables, XML, …)
- People pose queries in SQL, XQuery, …
  - Queries get routed to most efficient location(s) for query processing
  - Data gets routed as appropriate
  - Queries are processed, potentially at multiple sites, and information is returned to the user
- System makes efficient use of its resources
- Important data can move and be replicated

A Spectrum of Distributed Data Management Techniques

	Distributed Databases	**Data Integration**	**Wide Area Data Management**
Data	homogeneous	heterogeneous	heterogeneous
Data control	central	external	external
Schema	central	central	site-determined
Data sources	central admin	centrally mapped	ad-hoc, dynamic
Replication	manually specified	not a focus; limited caching	automatic

But this is a Problem with Many Heads!

Solving this problem requires:

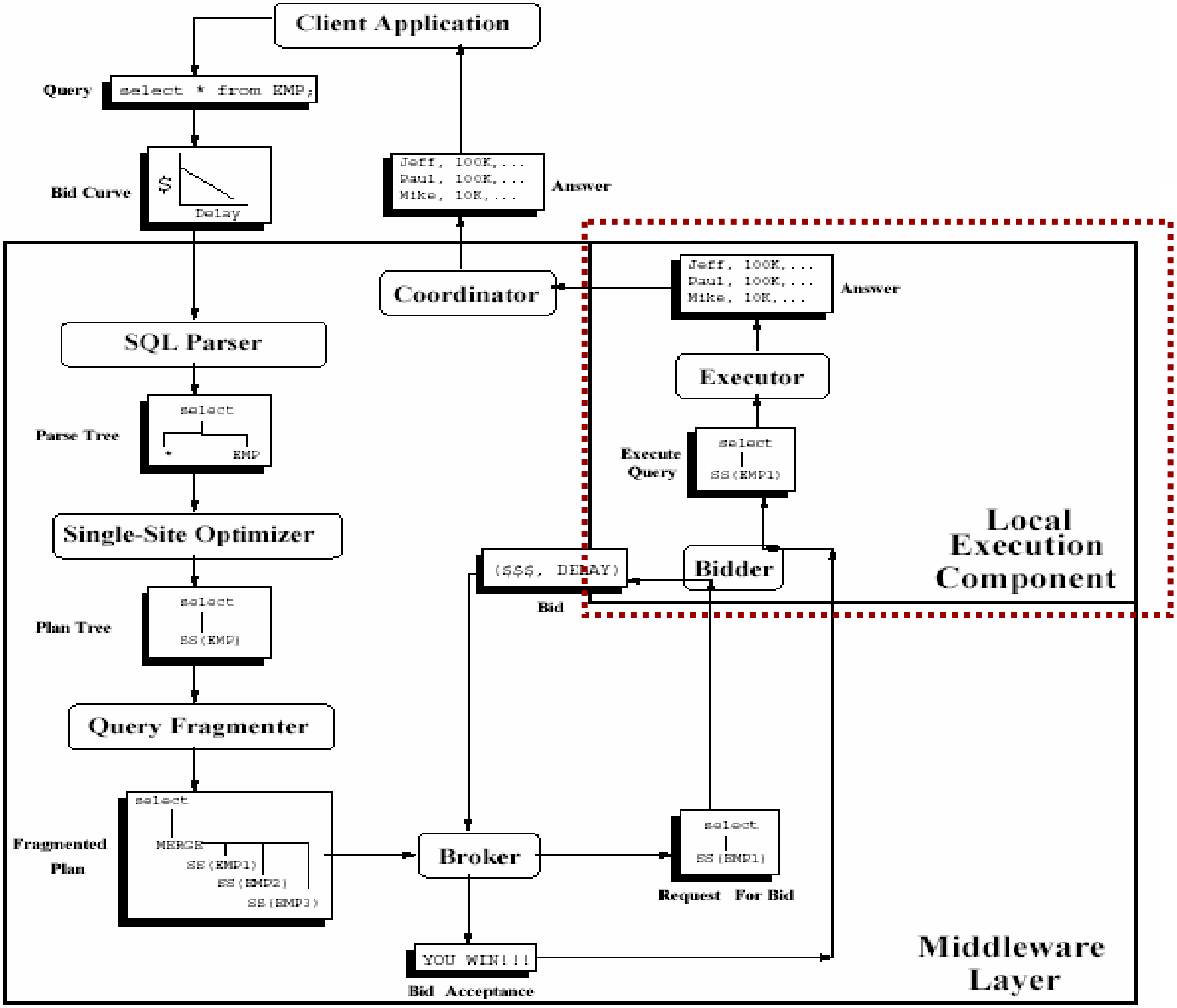
- Handling autonomy of sources
- Handling schema and data heterogeneity
- Handling scalability
- Providing performance
- **Providing a benefit that makes people want to use the system!**

Mariposa

- "Distributed DBMS for the wide area"
- Stonebraker projects: *gres, or sites of California Nat'l Parks (Sequoia, Big Sur, Mariposa, …)
- Goals:
  - Scalability
  - Multiple administrative domains
    - Autonomy of source policies
    - Autonomy of schemas, resource commitments
  - Data gets distributed to where it's in demand
  - Can negotiate for quality of service
  - Distributed optimization takes these factors into account

Core Idea of Mariposa

- Open markets – capitalism – works quite efficiently in matching buyers + sellers
  - Different buyers have different needs, demands
  - Different sellers have different resources, costs
- Use this model as the basis of resource allocation
  - Services have brokers
  - Participants (e.g., compute, data, storage providers) are sellers
  - Clients place bids


Client Application
Query
select * from EMP;
Bid Curve
$
Delay
Jeff, 100K,...
Paul, 100K,...
Mike, 10K,...
Answer
Coordinator
SQL Parser
Parse Tree
select
*
EMP
Single-Site Optimizer
Plan Tree
select
SS(EMP)
Query Fragmenter
Fragmented Plan
select
MERGE
SS(EMP1)
SS(EMP2)
SS(EMP3)
Broker
YOU WIN!!!
Bid Acceptance
select
SS(EMP1)
Request For Bid
($$$, DELAY)
Bid
Bidder
Execute Query
select
SS(EMP1)
Executor
Jeff, 100K,...
Paul, 100K,...
Mike, 10K,...
Answer
Local Execution Component
Middleware Layer

Mariposa Services

- Storage – buyers may want to store their data
- Data – the same data may be available in many places with different freshness levels
- Naming – data needs names & metadata
- Query execution – where does an optimized plan get executed?
- Brokers – match service providers with buyers via bidding

- Most of functionality governed by local “Rush” rules

Storage

- Can be:
  - Replicated in many places (with different guarantees)
  - Fragmented across multiple systems (vertical or horizontal partitions)
- Fragments can be split or coalesced as needed
  - (Never implemented?)
- Fragments bought and sold to maximize value

Naming and Finding Data

- Internal name = address (where an object is now)
- Full name = object ID
- Common name = user-specific alias
- Name context = a namespace

- Go to local cache, then go to name server
- Name server is a service and requires bidding
  - Polls various local catalogs
  - May have different QoS guarantees

Query Processing in Mariposa

- Distributed query optimization is REALLY hard
  - Need to try all combinations of executing different parts of the query on different machines
  - Regular optimization is already $O(3^n)$ or so…
  - So nobody really does full DQO
- Mariposa heuristic:
  - Optimize as if we're executing locally
  - Fragment the plan, break into strides (parallelizable)
  - Conduct bids on fragments

Optimization:
Bidding on Fragments

- For each computable fragment (each fragment in a stride), use one of:
  - Expensive bid protocol
    - Send out bid
    - Get back triples (Cost, Delay, Expiration of bid)
    - Notify bidders of winner
    - LOTS of messages
  - Purchase order protocol
    - Send to "most probable" winner (not clear how we know this)
    - Site returns answer + bill (no negotiation allowed)
- Heuristics to choose winners when many strides and bidders (e.g., consider each stride separately, use greedy algorithm to balance cost vs. delay)

Advertising and Pricing Services

- Here it's not clear what really got implemented…
- Service providers advertise in yellow pages
  - May publish rates
  - May need to provide "coupons" if overloaded
- Pricing is generally based on CPU and I/O resources
  - Can adjust by preference for certain data
  - Adjust by average load

Mariposa Wrap-up

- Contributions:
  - Interesting ideas about applying economic models
  - One of earliest systems to address wide area

- … But ultimately unsuccessful
  - System was never really deployed
  - Work ended by ~1997

Piazza: P2P + DB = PDMS (A Vision Paper)

Peer-to-peer has compelling vision but is limited:

- ✓ Build ad-hoc distributed system that scales via cooperation, resource sharing
- ✗ Simple data model and querying

New applications in data management if P2P vision used as inspiration

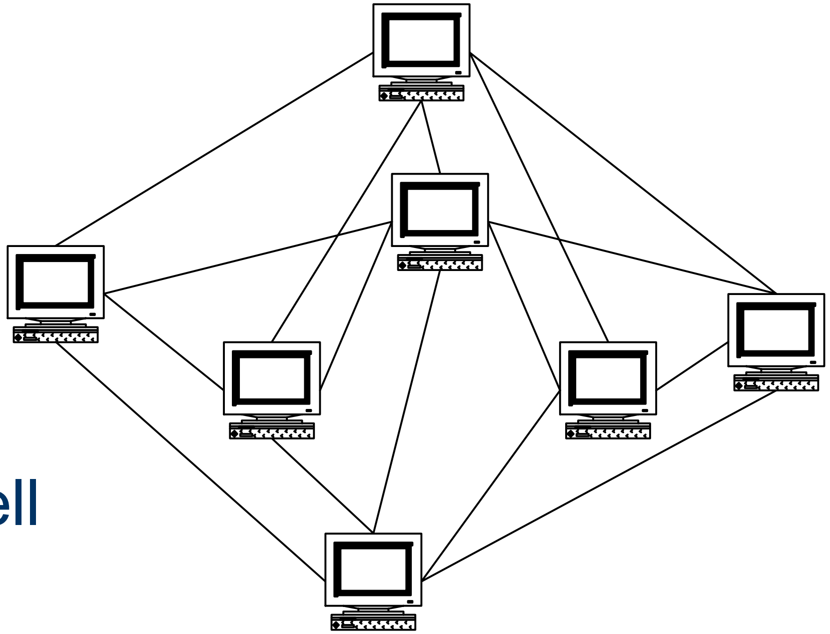
Example: data sharing for science

Goal of Piazza: P2P-like data management

Vision of Peer-to-Peer Computing

Benefits

- No central administration
- Scalability
- Adaptability/resiliency
- Nodes contribute as well as consume resources
- System continues as peers join and leave



Standard P2P: Missing Data Management

Focus: Cooperative storage and serving of files

- Napster
  - Centralized lookup
  - Scalable to limits of centralized directory
- Gnutella
  - High-overhead network protocols
  - May not find existing objects
- OceanStore [Kubiatowicz et al 00]
  - Global-scale persistent data storage across world
  - Designed for scalability

✗ No data model, primitive querying, ambiguous semantics

Extending the Vision Beyond Files

Suppose we added richer, DB-style semantics:

- Rich data & query model
- Schema mediation
- Peers provide query services (CPU resources)
- Peers materialize results (disk resources)

Imagine a Web where sites exchange semantically meaningful data

- Can answer much richer queries than today's Web
- Part of the "Semantic Web" (discussed later in the semester)

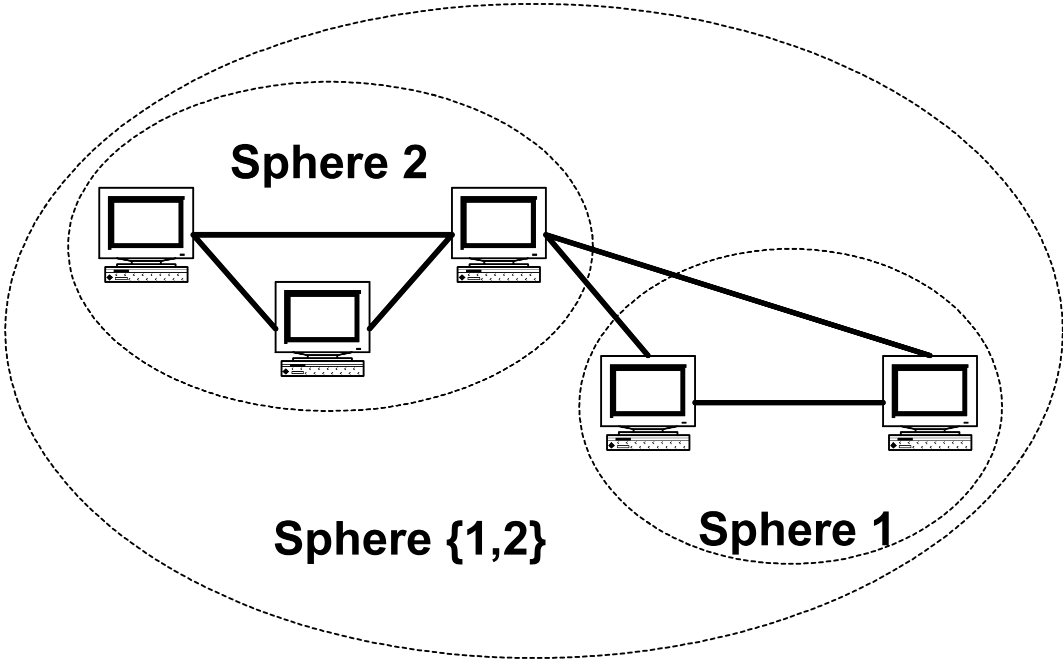
Piazza: Peer Data Management

Data management foundation

- XML querying
- Materialization of results where most useful
- Query optimization

P2P-inspired aspects

- Decentralized, ad-hoc
- "Spheres of cooperation": compromise between local and global

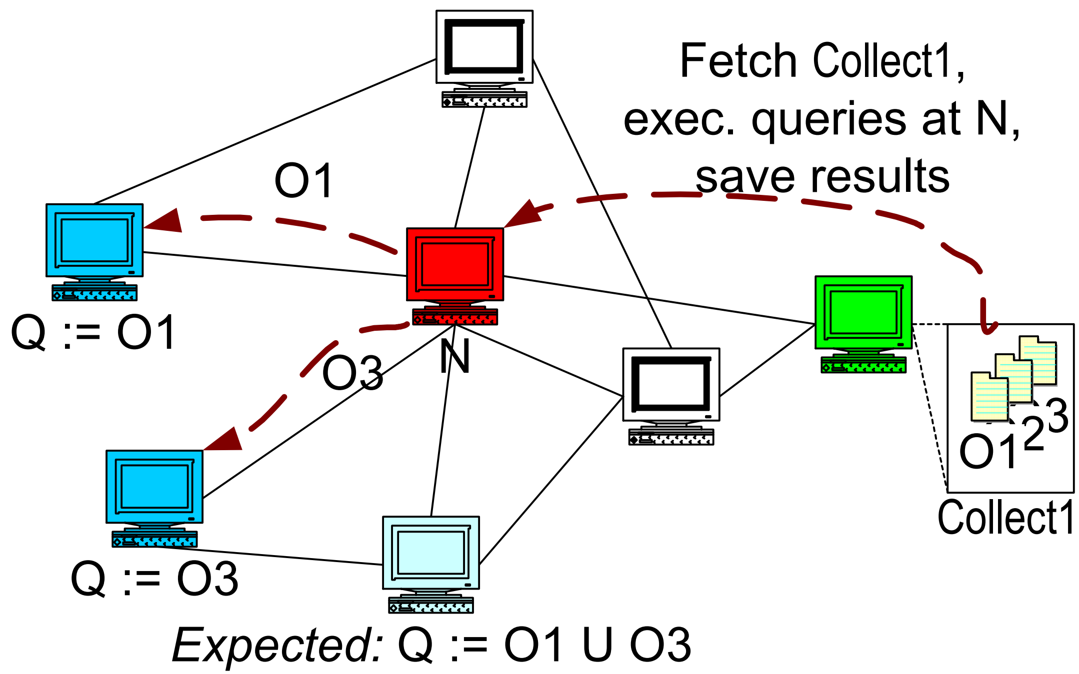


Initial Focus of Piazza: Data Placement

- Analogous to file replication in Gnutella
  - Results of a query become a “materialized view”
    - Answering queries using views!
  - Much more re-use possible with DB-style querying
- Problem:
  - Where do we place data so it can be maximally reused?
  - How do we answer queries while making use of this data, all in a scalable way?
    - Trade-offs between global and local decision-making

Optimal Placement of Data for Re-Use

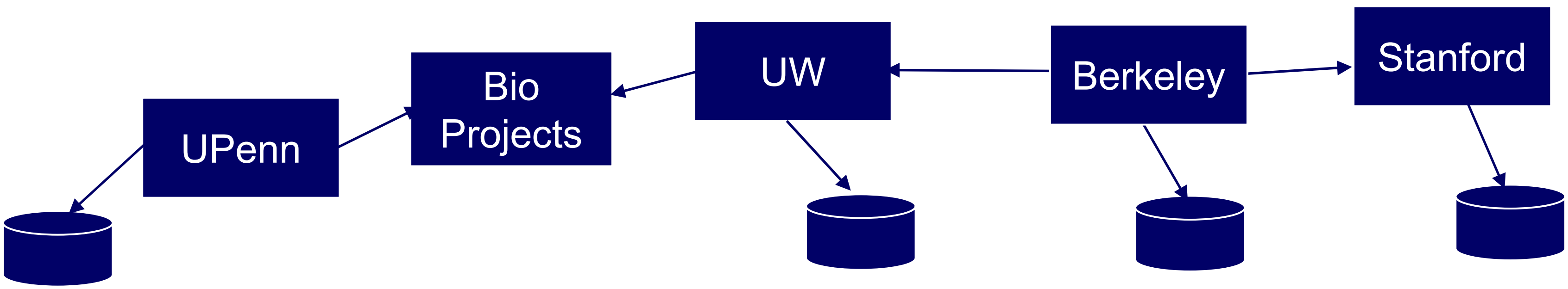
- After each query, decide where to place data for best performance
  - What to keep (materialize)
  - What to evict
  - How useful a query is if it overlaps



After the Paper: What Did We Learn about Data Placement?

- Can take many standard, naive algorithms
  - LRU, LFU, etc.
  - Can supplement them with a few other factors
    - How big is the data?
    - How often is it updated?
  - And can apply to either local nodes or to clusters of nodes
    - Compromise: spheres of cooperation – sites of similar interests
- How do we assess the results?
  - What's a typical workload?
  - First we need to understand how people use the system!
- Performance/scalability can't be assessed until we understand how a system is used!
  - We need a killer app!
  - This means we need a functional advantage!

A Functional Difference: Decentralized Mediated Schemas



- Each peer has own logical schema
  - Queries posed over specific version of this schema
- Mappings are created between schemas (or sources)
  - Like data integration – only everyone is a mediated schema
- Queries evaluated across chain of mappings

Discussion

- Key questions:
  - Mariposa: what can we learn from it?
  - Is Piazza destined for similar fate?
  - How many heads are on the hydra right now?

Why Did Mariposa Fail?

- The economic model is impractical
  - How do we price resources, bids?
  - How much money is in the bank?
  - Bidding takes too long
- Schema and data heterogeneity weren't addressed at all
  - Perhaps the #1 problem in distributed data sharing
- What application does Mariposa enable?

How We're Trying to Do Better in Piazza (The Jury is Still Out!)

- Try to drive research by building and deploying the system in real applications
  - Currently, simple data sharing applications
  - Hopefully: sharing biological data
- Heterogeneity is where we give benefits!
  - Decentralized mediation between large numbers of peers
- Part of a bigger-picture effort to facilitate semantically rich data sharing
  - In concert with semantic markup tools, semi-automated schema mapping, …
  - This is why we'll come back to Piazza a couple of additional times this semester…

Coming up…

- Query optimization – starting with a 24-year-old paper that's still relevant!
- Our first student presentation
- Guidelines for potential class projects