I. Background

Distributed databases:
- build an overall query plan using pieces for each site
- execute the pieces and then combine results
- all nodes agree on the schema
- nodes are supposed to execute whatever is asked for... not really autonomous

Federated systems:
- each site is autonomous
- can decline work, can even abort work
- some trust of others sites, but not complete
- local control over schema and allocation of resources

Questions:
- how to do a query over a federated system? one answer: take bids
- what do sites have to agree on to work together?
- what do you give up (in terms of autonomy) and what do you get?
- assumption: you benefit from using other sites for your queries, therefore you participate in their queries...

II. Principles

Data moves around:
- location info may be stale
- data may be stale
- focus on availability rather than strict consistency

No global synchronization
- eventual consistency for schemas, metadata

Scalability

Total local autonomy:
- control over what is stored
- control over which queries are run
Policies must be configurable
  o system provides mechanisms only; policies are in a scripting language (rush) and are easily changed

III. Economic Model

Key idea: establish an economic framework (with real or play money)
  o real costs vary widely
    • is the data local?
    • is the load low?
    • is there plenty of storage
    • network connectivity/cost
  o bid process should find nodes that can execute a subquery cheaply
  o don’t need to ask every site -- just get enough bids to get one good one
    • include sites that have the data, since they are likely to have low bids
    • can also seek more bids as needed
  o bid curve: tradeoff cost vs latency (quick results cost more)
    • e.g. if you just need the answer sometime today, price should be much lower
  o assumption: you need to bid on queries (presumably so that you can buy queries later)
  o broker handles the bidding process, coordinator handles the execution
  o purchase-order queries are not bid, just bought
    • lower overhead (no bidding)
    • may get a worse price, but mitigated by past experience with the provider
  o batch queries also reduce overhead and presumably price
    • easier to amortize costs if you have a whole set of queries to do

Fragments are parts of a table that are bought/sold
  o can also be split/merged locally
  o more fragments implies more parallelism but also more overhead

IV. Query Processing

Start with a single-site plan (as if all data is local)

Fragmenter creates as “fragmented query plan” based on the distribution of fragments
  o also defines “strides” which are multi-fragment queries that execute in parallel

Broker: bids out each of the subqueries
  o bids contain [site, cost, delay, expiration of offer]
o looks at the collection of bids for each stride independently (since strides execute sequentially)
  • subqueries within a stride execute in parallel
  • hard to estimate the total delay, but just use the max
o starts by picking set that minimizes the delay (but not the cost)
o then switches some subqueries to reduce cost (but add delay)
o keep a switch if the result leaves more room in the bid curve (greater difference between cost and bid curve)
o not optimal, just a heuristic
o notifies winners (notify losers too, although they can just expire their bids)

Coordinator: execute the queries

Site view: must compute a bid
  o simple version: real cost * (safety factor) * (load average) + cost for missing pieces

V. Name Server

Need to track which sites have which fragments, and willingness to bid

Uses advertisements to find out which servers are willing to perform which tasks
  o “yellow pages” ad: specific service and specific price (subject to change)
  o Can post a “sale price” or use “coupons”
  o Bulk prices also important
  o Broker can use ads and past experience to solicit bids

Name servers need to be paid too -- could be by the broker or by the seller

Freshness of data is a metric for name servers -- fresh data should cost more

VI. Storage

Need to decide what fragments to store

General answer: buy those fragments that would have helped you win past bids
  o keep of list of hot fragments that affected revenue
  o when one of these comes up in a bid, consider buying it
  o net present value = discounted value of future cash flow - investment
    • typically a sum over n future periods, where the discount rate for period i is r^i, so periods far in the future are given little credit
  o can buy on demand (but must factor in delay), or can prefetch the fragment (but you may not win the bid)
may have to evict fragments, ideally by selling them

VII. other questions

How does this fit with P2P? OceanStore?
Do you trust all other participants or just some?
Reputation?
Penalties?