

# CS286: Database Systems

## 1 Lecture 3—9/4/2014

### 1.1 R\*

- Assumptions:
  - There are administrative causes behind distributed data
  - Network: unreliable transport, in-order, packets are intact
  - Independent node failure
  - Slow-ish network
- Research goals:
  - “Site autonomy”: No centralized state or control
    - \* Data you touch should determine the sites you talk to
    - \* “Distributed system is a system that fails because a machine you’ve never heard of fails”
    - \* Load sharing and decentralization
    - \* Less communication
    - \* Harder to coordinate data consistency
    - \* More network connections beyond hub and spoke
    - \* Metadata management is harder
  - Location transparency → emulate a centralized DB
  - Don’t assume much about the network or OS
- Highlights:
  - Query optimizer cost modeling
  - Data layouts → horizontal partitioning
  - Replication
  - Distribution
  - Query compilation—unclear as to balance between compilation overhead and work saving
  - Spent a lot of time talking about 2PC → presumed commit

### 1.2 Gamma

- Assumptions:
  - Fast interconnect—hypercube, more network bandwidth than aggregate disk bandwidth
  - Shared nothing—no disk or memory sharing
- Research goals:
  - Scale

- Highlights:
  - Parallel hybrid-hash join
  - Chained declustering
- Assess:
  - Linear speedup + scale-up
  - Superlinear speedup due to minimized seek count at scale

## 2 Lecture 4—9/9/2014

- ACID
  - Consistency is not what we typically think
  - Distributed systems: data has a consistent value across sites
  - Databases: data meets contract when transaction completes
- Serializability mathematically gives atomicity and isolation
- Logging gives atomicity and durability
- Ordering:
  - Determines outcome (unless operations are not associative and commutative)
  - Some things are commutable/associable
  - Ordering must be equivalent to some serializable order
  - Implicitly, this provides an API—people don't *need* to reason about concurrency
- What is storage?
  - *Spacial-temporal rendezvous makes everything work!!!!*
- Want to avoid/undo conflicts in space and time
  - *Space*: Shared names
  - *Time*: Ordering
- 2PL: Provides a conflict serialized schedule
  - Ordered by race for locks
  - Ordered by the end of the first phase (“lock point”)
- Multi-version timestamp ordering
  - Every transaction gets a timestamp—this is the only synchronization point
  - For every object:
    - \* Writes generate a new version for an object
    - \* Reads annotate the version for the object