Designing Data-Intensive Applications

Chapter 12: The Future of Data Systems

Chapter Overview:

- Authors prediction for the future of data systems
- Trying to wrap up how to bring things together for reliable, scalable, maintainable systems

Data Integration

- We have looked at various different technologies, but there is no one right solution
- Need to evaluate tools from various different vendors with the knowledge from what we have learned with a critical eye for what vendors might be intentionally omitting.
- Might be the case that no one technology matches all of your needs and multiple need to be used in conjunction

Combining Specialized Tools by Deriving Data

- Some features might not be useful to one organization but are useful to another. It is valuable to understand all of the various ways that data might be written and read
- There are trade-offs between write complexity and read complexity
- There are different constraints that might be needed for each (eventual consistency, read on write, etc.)

Batch and Stream Processing

- Goal of data integration is to make sure data ends up in the right form in the right place
- Often have to create derived dataset for easier read operations separate from initial write operations
- Strong new contenders are pushing in the direction of systems somewhere in between batch and stream processing
- Derived views allow gradual evolution
- Lambda approach uses a stream processor to make approximate updates and then batch processor later produces a corrected derived version
    - Maintaining both batch and stream code at once keeping them synchronized can be cumbersome

Unbundling Databases

- Unix and relational databases have approached problems from a very different perspective
    - Low level abstraction vs high level abstraction that hides many details, but adds optimizations

Composing Data Storage Technologies

- Proposes two futures of data solutions, Federated databases: unifying reads, Unbundled databases: unifying writes
    - Unified query language that pulls information from a wide range of technologies automatically. Can pull data from multiple different types of stores and aggregate them using a sort of meta query language
    - Unified writes across technologies, unbundling databases index maintenance features so that writes can be synchronized
- Still a complicated problem that needs to be solved, but ultimately the mapping between technologies isn't impossible because many use somewhat similar techniques

- Pushing things towards event logs instead of transactions can make some of this more compatible and more robust with greater independence between systems
- Goal of unbundling isnt to compete on performance, but to allow more flexibility and combine the advantages of different technologies to allow data to be consumed in many ways (breadth over depth)
- Currently missing unix like commands that can pipe one technology into another

Designing Applications Around Dataflow

- Spreadsheets have dataflow capabilities that are far beyond what we have in databases currently. Automatic recalculation when formulas and data change all abstracted from user
- Very often the database can do its built in indexing functions well but custom functions are where things are slowed down
- Probably makes sense to have some systems focus on durable data and others that specialize in running application code for more custom functionality
- Most of current data solutions built with web in mind where stateless is assumed
- Future is possibly pushing toward more active state management, application code responds to state changes in one place and triggers changes in another
- Exchange rate example
    - System might query exchange server everytime a purchase is made
    - Alternate approach is to have any exchange rate changes pushed to the server so no outbound requests are required
- Materialized views and caching, can process nothing-everything beforehand and trade read path vs write path
- Many web systems starting to become stateful and might break from the request response paradigm, can still do useful things even when offline
- Can receive updates after regaining connection, publish/subscribe

Aiming for Correctness

- Lots of applications moving away from assumptions of consistency
- Consistency guarantees might fall apart at high concurrency anyway

The End-to-End Argument for Databases

- Even with database safety guarantees, it is still possible to write incorrect data to DB
- Can use operation identifiers to make operations survive through the whole write procedure and make sure not duplicated

Enforcing Constraints

- Might want to make sure accounts dont go negative or sell stock you dont have, etc.
    - Can be complicated, but possible with log based messaging that requires confirmation of operations
    - Transaction example - Can log a single request and have it derived the credit and debit instructions from the single instruction

Timeliness and Integrity

- Different definitions of timeliness relevant for different use cases
- Integrity is absence of data corruption
- Integrity commonly more important than timeliness
- With loose constraints can often just use existing processes to make up for any constraint violations, sending emails saying a username is already taken, etc.

Trust, but Verify

- Important to have some form of auditing to actually verify data is correct
- Also important to make sure backups are actually functioning
- Various crypto techniques emerging that solve some of these problems, but still be developed

Doing the Right Thing

- Have to consider the ethics of data, its storage, and its usage
- Possible that used in a biased way or for discrimination
- Sometimes data used in opaque ways that cant be queried in any way like a human decision could
- Privacy and tracking are a big concern, author proposes the exercise of swapping the word data for surveillance and considering implications
- Sometimes it is obvious what the user is willing to share in terms of data but sometimes it is a bit more gray given what an app can infer from usage and various other datastreams that are accessible