Finding a Good Home

Choosing the Right Data Store

Jeff Darcy

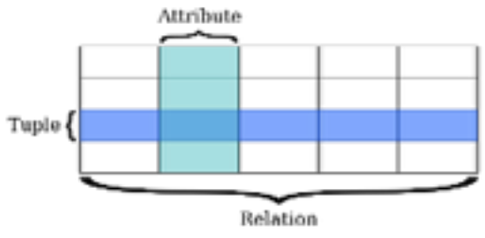
LISA'14

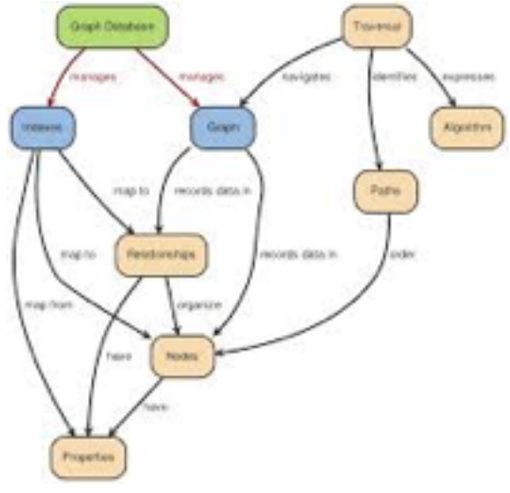
Overview

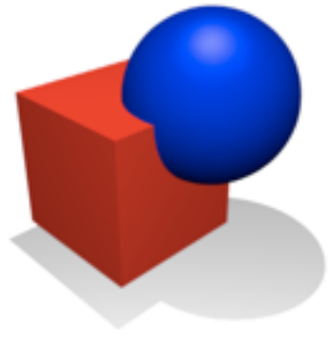
- Introduction, conceptual framework
- Physical layout
- Logical layout and semantics
- Impact of new technologies
- Recommendatons throughout

The Storage Matrix

Physical ✖ Logical







Physical: Local Disk

- Includes server-resident flash etc.
- PRO: simple, cheap, no coherency problems
- CON: finite capacity, performance
- CON: doesn’t address server failure
  - need your own heartbeat/failover + replication

Physical: SAN

- PRO: reliability, in-array functionality
- MIXED: performance, capacity (if you have $$$)
- CON: still doesn’t address server failure

Physical: NAS

- Basically same pro/con as SAN
- Different cables and protocols don’t change that much
    - slightly different emphasis on networking vs. storage skills

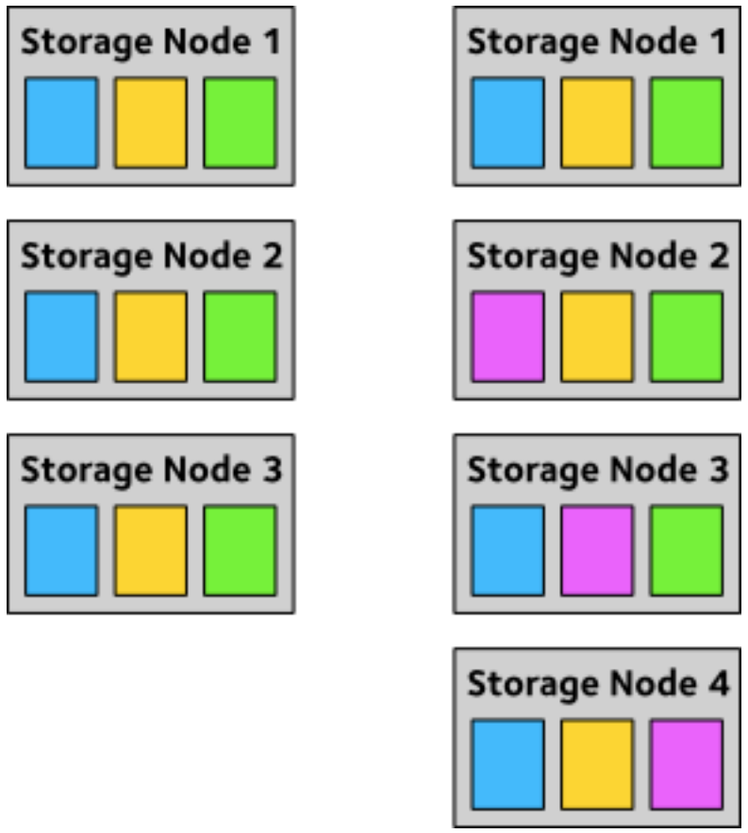
Physical: Distributed

- Includes file systems, databases, object stores
  - also “server SAN” and “hyper-converged” 😝
- PRO: scalable, cheap (if open source)
- MIXED: modern designs do address server failure (just say **no** to metadata-server SPOF)
- CON: high operational complexity, lots of “dark alleys” w.r.t. performance and maintenance

Replication and Consistency

Inspiration from Kyle Kingsbury (@aphyr)
Really, go read
http://aphyr.com/posts/281-call-me-maybe-carly-rae-jepsen-and-the-perils-of-network-partitions

Rebalancing

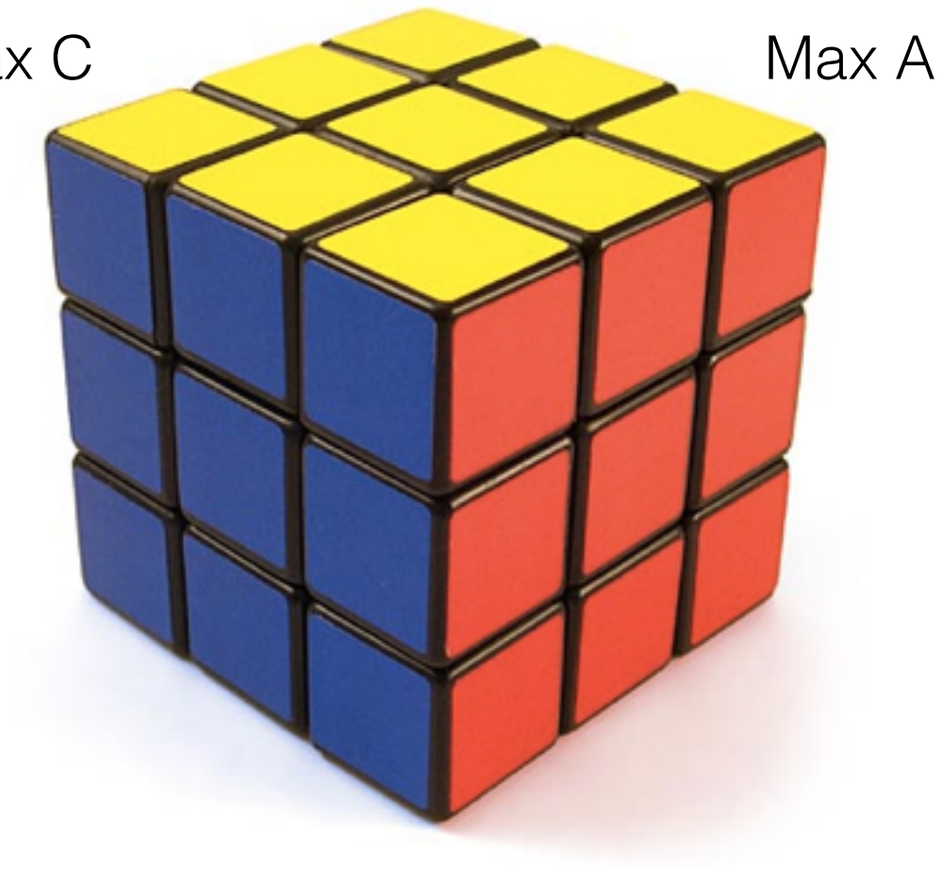


http://www.drbd.org/users-guide-9.0/s-rebalance-workflow.html

CAP Cube

Max C

Max A



Max P

- Consistency, Availability, Partition Tolerance
- “Pick two” is misleading
- Still can’t have maximum of all three (can’t go around the back)
- Must pick your tradeoff(s)

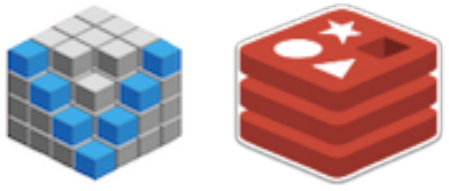
Recommendations

- Use local storage whenever you can
  - newer technologies might only be consumable that way
- SAN/NAS : not dead yet (obMontyPython)
- Do be wary of distributed storage
  - It can be great, often necessary, but it can also be the greatest source of pain in your life

Logical: Relational DB

- ACID is great
- Multiple/compound indices, ad-hoc queries, sophisticated query planners → also great
- Inflexible formats, normalization, difficulty adding columns → not so great
- Most have good replication/scaling stories nowadays

Logical: NoSQL



Key/Value

Document

Columns

Graph

Logical: NoSQL

- Many different data models, consistency models, performance profiles

- Very generally:
  - Good scalability
  - All of the standard distributed-system problems (but furthest ahead on solving them)
  - More “dark alleys” mostly due to unfamiliarity and new semantics (eventual consistency, vector clocks, CRDTs, …)

Logical: File Systems

- Disclaimer: this is my own area, so of course I'm biased
- Surprisingly similar to distributed databases - same algorithms, same tradeoffs, same problems - but different semantics
- PRO: high familiarity/compatibility/applicability
- CON: very hard to implement some features correctly and with good performance - e.g. byte-level overlapping writes, O_APPEND, cross-directory rename

Logical: Object Stores

- One- or two-level hierarchy, get/put whole object, no rename, etc.
- Simpler semantics supposedly make these more scalable than file systems … O RLY?
  - You still have to deal with hard problems (e.g. repair/rebalance)
  - Users have to deal with lack of partial updates, security, **consistency**, …
- If you use the same subset of FS functionality, you get almost the same simplicity without the sacrifices

Recommendations

- Use databases if you need many searches and queries
  - pick a type based on specific consistency needs, “natural” data model, performance profile
- Use file systems for compatibility, or for hot/warm data accessed via known paths
  - disk images fit this model well

Tech: Erasure Coding

need k/n, arbitrary k and n

- Uncommon now, ubiquitous in next couple of years
- Great storage efficiency, durability
- Terrible performance (especially rebuilds)

Tech: Phase Change

- Inverted read/write performance vs. flash
  - Flash: 108µs read, 37.1µs write
  - PCM: 6.7µs read, 128.3µs write
    - (https://www.usenix.org/system/files/conference/fast14/fast14-paper_kim.pdf)
- Density not quite equivalent to flash
- Other concerns: power consumption, temperature sensitivity

Tech: Other NVM

- FeRAM: fast (60ns), lower density even than PCM, power efficient, fatigue/imprint
- MRAM: even faster (35ns), density and power consumption similar to FeRAM
- Further out: Racetrack, SONOS, others
- Various groups working on new APIs etc.

Tech: Spinning Disks

- Increasing density: HAMR, patterned media, helium
- Shingled Magnetic Recording
  - overlapping tracks allow higher density
  - great for read and sequential write, rewrite becomes <u>really</u> expensive
  - conventional file systems etc. slow to catch up (irony: log-structured FSes designed for other media might be revived for SMR)

Layering

- Hybrid devices (e.g. nvDIMMs)
- Explicit tiering across devices (or servers)
  - MRAM + flash + normal disk + SMR in one system
  - Also tier based on encoding, power, durability
- All magic, until you cross one of those tier-size boundaries
- Manual data-wrangling will drive you insane

Recommendations

- Don’t be afraid to mix and match, pair storage types
  - tiering, segmentation by type, DB+files
- Have a plan (and tools) to manage data across more storage types/layers than you’ve ever had before

Questions?