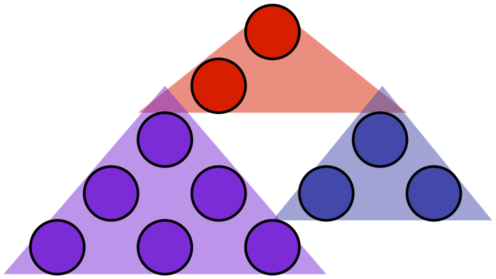
Posix-Free File Systems in the Cloud

Jeff Chase

Duke University

Beyond Posix

- "Filesystem" ➔ Posix file system semantics?
  - open(2)
  - Hierarchical directories with aliasing
  - Human-readable symbolic names
  - Atomic ops on directory tree
  - Consistency, etc.....
- It has served us for more than 25 years...

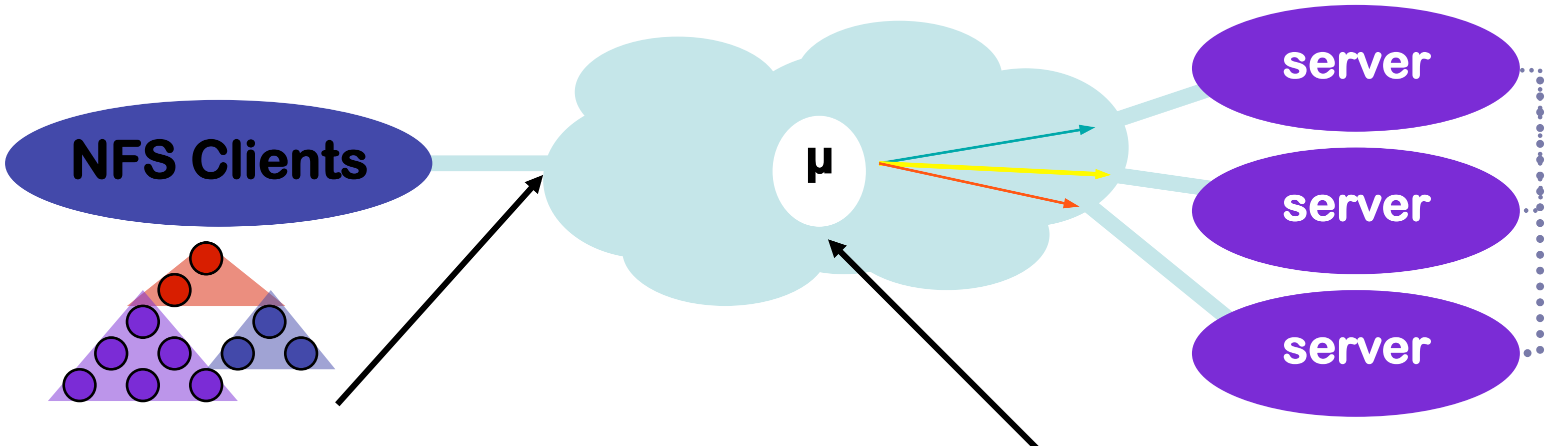


Continuum of File/Storage Systems

- Personal devices
  - Small apps, common file system
  - Seltzer and Murphy, Hierarchical File Systems Are Dead, HotOS 2009.
  - Do you know where *your* files are?
- Server backbone
  - Your data lives here; devices are caches.
  - Storage sits behind client-facing apps
  - Big $$$ apps and infrastructure
- Server storage is breaking out of the straitjacket.

SLICE

[OSDI 2000, TOCS 2002, USITS 2003]



Client sends and receives standard NFS.

Storage router module intercepts and redirects requests to servers.

rename()!&*^%

Server "File Systems"

- Trend: storage abstractions as foundational services.
  - Robust, scalable, etc., etc.
- Google FS (GFS SOSP 2003)
  - "Co-designing applications and the file system"
    - FS tailored to workload (large files)
    - Apps program to "new" storage API
    - Apps compensate for quirks of FS
      - E.g., record repair at application level

"Have it your way"

- Now evolving toward a rich menu of more specialized storage APIs with features to fit.
- Key-value stores
  - Amazon S3, FAWN, etc.
- Multi-attribute indexing (tables or property lists)
  - Amazon SimpleDB, Google BigTable/Megastore
- Content-addressable
- Temporal/lifecycle management
- Etc.

Into the Clouds

- Cloud == "data center consolidation"
  - Pay as you go
- Diverging views of storage in the cloud…
  - Cloud of public services
  - Cloud of public virtual infrastructure to host private services
    - E.g., GENI
- These choices lead storage system design in different directions.

Some key differences

- Accounting must be “designed in” to public services.
  - (unless they’re free)
- Trusted platform vs. trustworthy services
  - Public services need data protection (whatever that means to the customer).
  - E.g., strong accountability (FAST 2007)
- Elasticity
  - Public services need some kind of isolation…
  - For private services, elasticity ➔ churn
  - Controllable (re)scaling and data (re)placement

Other…

- Data/vendor lock in with the public service model?
  - Unless we standardize storage APIs
- How to expose/manage location?
- How to expose/manage device properties?
  - Encapsulate at bottom layer?
- Risk of feature-creep for public services
  - Snapshots, cloning, etc.
  - “Stackable” storage services?
- How much customization do we need?
  - One size fits all vs. let a thousand flowers bloom

Storage Software as a Service

- Cloud provider runs common storage services shared by multiple customers.
  - Thin straw problem? Your application is in the cloud too.
  - Beware: data lock-in, one-size-fits-all
- The storage service must have designed in:
  - Elastic scaling with performance isolation
  - Data protection (whatever that means)
  - Accounting (unless it's free)
  - Accountability

Infrastructure as a Service

- "Infrastructure as a Service" model
  - Instantiate virtual machines and virtual devices
  - Let a thousand flowers bloom
  - Example: GENI
- The storage service must have designed in:
  - Controllable (re)scaling and data (re)placement
    - Elastic ➔ churn
  - How to expose location?

GENI Storage

- Decouple services from infrastructure
  - Common “raw” sliverable storage infrastructure?
  - “Let a thousand flowers bloom.”
- Consider separate services separately
- Focus on key storage services for workflow
  - Repositories: Image/appliance, snapshots, source (?)
  - Operational: auditing, instrumentation (write-once)
  - On-demand storage for experiment use
    - Node sliver instantiation (roots)