Cloud Core Services

Jeff Garzik
Principal Software Engineer,
Red Hat
Inspirations: clouds move to production

• Dawn of new distributed computing era, based on production experience with massively scaled clouds and open Internet protocols, such as at Google and Amazon.

• Microsoft Boxwood research from 2001 inspired some Google work, notably PAXOS consensus algorithm.

• Google's publishing of papers on GoogleFS, Map/Reduce algorithm, Chubby (lock service) and BigTable illustrate how a few key userland services, linked into applications via client library, may collaborate to provide high performance, fault tolerant services.

• Amazon S3 illustrates an easy, standards-based (HTTP REST) service that enables its users to easy built powerful, infinitely scalable [in terms of data storage] web applications.

• Generalized end-user application needs: Reliable, scalable turnkey services for developers to use in building their own applications. Analogy: using a database library (db4, sqlite) or service (postgres, mysql) rather than writing it yourself. In this context, we see low-level service components such as
  – Distributed BLOB data storage
  – Indexing, such as distributed key/value map (hash, b-tree, ...)
  – Consensus and coordination (leader election, synchronization, ...)
Building basic cloud data services and APIs

- Cloud computing well suited to Linux (Beowulf history; existing Red Hat efforts in virtualization and grids)
- Enable customers to build their own clouds easily
- Provide language-neutral basic commodity cloud building blocks, via an open network protocol:
  - Base node coordination (coarse locking, name service)
  - Advanced node coordination (producer/consumer queues and mailboxes). Perhaps AMQP can help?
  - Infinitely scalable lookup table (Unlimited [key, value] storage)
  - Unlimited-size 2D table (spreadsheet)
  - Indexing (fast searches)
  - Map/reduce: algorithm for distributing calculations across a cloud
  - Hooks into existing cluster management systems, for admin notification, hardware addition and removal, requesting new nodes for a service etc.
“Project Hail”

- Home page: http://hail.wiki.kernel.org/
- Core services
  - “CLD” - coarse locking service. Provides consensus, master election, name space and critical file storage.
  - “chunkd” - chunk service, storing and retrieving anonymous BLOBs
- Example cloud application
  - “tabled” - infinitely scalable data storage via a lexicographically sorted(key, value) lookup table, compatible with Amazon S3 client using standard HTTP REST network protocol.
CLD's API – what you expect of a filesystem

• Methods
  – Open (mode read/write/lock/ACL, name, event mask, lock delay, initial contents), Close
  – Get (metadata + data, or metadata only), Put
  – Delete
  – Lock, Unlock, Try Lock
  – Get Lock ID, Set Lock ID (used for informing a third party of a lock)

• Asynchronous Events
  – Contents changed
  – Child node (sub-directory) added / removed
  – Master fail-over
  – Handle invalid
  – Lock acquired
  – Conflicting lock request from another client
“Project Hail” future

• Additional core services

• Additional applications: production-ready, useful examples
  – Distributed NFSv4
  – Distributed block device via iSCSI
  – Userland distributed filesystem?

• Expand client language support for core cloud service to Python, Java, C++

• http://hail.wiki.kernel.org/