Managing Massively Distributed Replication (work in progress)

i-CAN summer school and workshop
Tuesday, June 2nd 2015, Athens, Greece

Pekka Nikander
Research Fellow, Aalto University
(and CTO, Solu)
Outline

• Introduction
• Information Centric Networking
• Cloud computing
• Key concepts
• Back to the first principles
• Summary
Introduction

• Speaker background

• Goals

• A teaser: a glimpse of what to expect

• Note: Best new results often are found in the cross section of two or more fields
Speaker background

• Dr. Pekka Nikander, Aalto University
  • Also CTO, Solu, a startup company
• Developer and architect since 1980
  • Still writing code almost every week
• PSIRP co-inventor, with Dirk Trossen
  • PSIRP → PURSUIT → POINT
Goals

• Cross-section of ICN and cloud, or
• Whither to network for social (web) apps?
  • Cloud grew from Web apps
  • ICN started from the observations of traffic duplication (and security issues)
  • How to build an “elastic network”?  
• This presentation: an ICN vs KVS view
Higher-order goals

• Concerns about data ownership
  • Who “owns” my data in the cloud?
  • How to build a neutral platform?
• Could we move computation back to edge?
  • While keeping primary storage in cloud?
• How to secure the data?
<table>
<thead>
<tr>
<th>Similarities and differences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ICN rendezvous</strong></td>
</tr>
<tr>
<td>• Name → Content</td>
</tr>
<tr>
<td>• Scale to trillions</td>
</tr>
<tr>
<td>• Decentralised, secure</td>
</tr>
<tr>
<td>• Mostly static data</td>
</tr>
<tr>
<td>• Typically one writer</td>
</tr>
<tr>
<td>• Consistency ignored</td>
</tr>
</tbody>
</table>
Information Centric Networking

• Assuming you already know ICN basics
• Quick recap of idICN
• Some basic observations
idICN
[Fayazbakhsh et al, 2013]

- ICN principles and benefits
- Cache placing & routing measurements
- Basic architecture
Principles

- Decoupling names from locations
- Pervasive caching / massive read replication
  - Silently assumes stable data
- Often with nearest replica (NR) routing
- Binding names to intent
Benefits

- Lower (read) response latency
- Simplified traffic engineering
- Security, mobility
- Ad hoc mode support
Caching results

- Assumes scale free tree-like network
  - (Rocketfuel topologies; Spring et al 2004)
- Assumes Zipf data distribution, varying $\alpha$
- Results (simplified):
  - NR routing helps 2–17% vs EDGE only
  - Simple extensions narrow this down
Consequence: idICN

- Incremental design, using existing protocols
- Self-certifying URLs
  - label.principal.idicn.org
  - principal = hash(public key(s), …)
- HTTP caches, automatic config (WPAD)
- Metalink-based metadata in requests
Content-oriented security can be provided via:

1. **Automatic configuration**
   - Proxy requests new names from the Name Resolution System.

2. **Content request by name**
   - Client requests content by name.

3. **Name resolution**
   - Name Resolution System resolves the name to an address.

4. **Content request by address**
   - Reverse Proxy requests content from the Origin Server.

5. **Routing the request and receiving the response**
   - Origin Server responds with the content.

6. **Response along with metadata**
   - Reverse Proxy sends the content to the client.

7. **Response**
   - Client receives the content.

**Key Components**

- **Origin Server**: Publishes newly generated contents.
- **Reverse Proxy**: Sends requests to the Origin Server.
- **Name Resolution System**: Resolves names to addresses.
- **Client**: Requests content by name.

**Figure 11**: A high-level view of the idICN operation.
Observations

- Name resolution is the crucial thing
- How frequently can names be updated?
  - Frequent updates gets complicated
- Rendezvous inside the network is the other
- Cryptographic names do help with security
  - Been around since DONA, at least
Outline

• Introduction
• Information Centric Networking
• Cloud computing
• Key concepts
• Back to the first principles
• Summary
Cloud computing

- Cloud computing in general
- Key-Value storage
- Key concepts
Cloud computing
[Armbrust et al (UCB), CACM 2010]
Cloud economics aka use cases

• Demand for a service varies with time
• Demand is unknown in advance
• Batch computation such as analytics

• I.e. non-uniform distribution of demand
• Pay-as-you-go or usage based pricing
Cloud novelties
[Armbrust et al (UCB), CACM 2010]

- Elasticity …
  - Infinite computing on demand
  - Elimination of an up-front commitment
  - Ability to pay for use a short-term basis
- … resulting from economies of scale and statistical multiplexing
Cloud application model

- Stateless model of computation
- Replicas instantiated on need
- Stateful storage tier
  - Our main focus today
- Virtual communication links
  - Fully anonymous nodes
Some cloud issues

• Security, especially *decentralised security*
• Data transfer costs (tens of $ / terabyte)
  • E.g. in EC2 outgoing traffic most expensive
• Variation in net and disk I/O performance
• Elastic, robust, consistent, scalable storage
Cloud storage

- Disk storage aka legacy emulation
  - GFS, Hadoop, Ceph, Lustre, IBM GPFS, …
- NoSQL databases
  - KVS, wide column, graph, object, document, …
- Focus: Elasticity, resilience, consistency
Well known KVS systems

- Dynamo (Amazon)
- Riak (Basho, Open source)
- LevelDB (Google)
- HyperDex (Cornell)
- Project Voldemort
Other well known ones

- Wide-column databases
  - Cassandra (Apache, Facebook)
  - BigTable (Google) and descendants
- Document stores
  - MongoDB
  - CouchDB
Outline

• Introduction
• Information Centric Networking
• Cloud computing
• Key concepts
• Back to the first principles
• Summary
Some key concepts

• Consistent hashing
• Paxos consensus
• CAP theorem
• Consistency models
• Conflict-free replicated data types
Consistent hashing
[Karger et al 1996]

• Goal: Elimination of hot spots

• No need for users have current or even consistent view of the network

• Adding or removing a node (hash bucket) requires only $K/n$ keys to be relocated

• Each node is mapped to several locations at a typical DHT algorithm circle
Consistent hashing

A ring with 32 partitions

$2^{160}$

$2^{160}/4$

$2^{160}/2$

a single vnode/partition

hash("artist", "REM")

node 0

node 1

node 2

node 3
Paxos consensus
[Lamport 1998, 2001]

- Optimal protocol for guaranteed consensus
  - Either majority agrees or fails
- Two phase: setup and consensus agreement
- Only $n/2+k$ messages at agreement phase
  - In non-failure case; theoretical minimum
- Several variants; some practical complexity
- Chubby, Spanner, Megastore; AWS; Ceph, …
CAP theorem
[Brewer 2000, Gilbert & Lynch 2002]

- Consistency
- Availability
- Partition tolerance

- Theorem (misunderstood): Pick any two
  - Better: (re)define consistency
Consistency models

• History of consistency
• Data-centric consistency
  • Mostly ignored in this talk
• Client-centric consistency
• Eventual consistency
• KVS consistency models
History of consistency
[Vogels, 2009]

- 70s–90s goal: distribution transparency
  - Better to fail than be inconsistent
- Then came CAP (2000, 2002)
  - Relaxing consistency for availability
- Business models favours availability to consistency
- Today mostly weak consistency; BASE
  - Basically Available, Soft state, Eventual consistency
Client-centric consistency
[Tannenbaum et al 2007; Vogels 2009]

• Eventual consistency; e.g. the DNS system
  • Monotonic writes (very hard without this)
• Monotonic reads
• Read your writes; session consistency
• Writes follow reads
• Usually you want monotonic reads & session
KVS consistency models

• Many have tunable consistency
  • E.g. eventually & strongly consistent reads
• Typically consistency within a ‘section’
• Many opt for always available writes
  • Often last write wins; read repair
CRDTs
[Shapiro and Preguiça, 2007]

- Data structures that achieve both
  - strong eventual consistency (SEC) and
  - monotonicity (in absence of rollbacks)
- Conflicts are mathematically impossible
- Used e.g. in Riak (⇒ League of Legends)
• Assume a merge function
• No matter where a state change is made
• Change will eventually make to all replicas
Back to first principles

- Usage or traffic model
- Information model
- Consistency revisited
Usage or traffic model

• 10 years ago web was mostly information consumption
• Same static content published widely
• Today web is mostly apps that tailor content and implement “transactions”
• Fetch and store KVS and other data
• Majority still static content, but how long?
Information model

• ICN assumes a prepare → publish cycle
• Documents are becoming more dynamic
  • Updates, comments, collaborative editing
• Multiple legitimate simultaneous versions
  • Development vs production, A/B testing, …
• Backups, version history, …
Consistency revisited
[Vogels, 2009]

- \( N \) # of nodes that store replicas
- \( W \) # of replicas to acknowledge update
- \( R \) # of replicas contacted on a read
- \( W+R > N \) \( \Rightarrow \) Strong consistency
- Stickiness strengthens weak consistency
- Application assistance needed for merges
Summary

- *Consistency* is often-ignored bastard in ICN
  - Weak eventual consistency may be fine
  - For interactive web apps typically not
- *Decentralised security* is a step child in clouds
  - Most assume crunchy shell, soft core
- Cross-field areas are often unchartered!
  - But better remember the first principles
References


