Volley: Automated Data Placement for Geo-Distributed Cloud Services

Authors:
Sharad Agarwal, John Dunagen, Navendu Jain, Stefan Saroiu, Alec Wolman, Harbinder Bogan

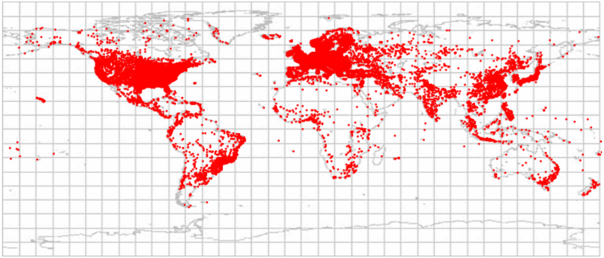
7th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2010)

Martin Beyer

December 14, 2010

Cloud Services

- Users from all continents want to collaborate through cloud services
  - Do not accept high latencies
- Cloud services deal with highly dynamic data (e.g. Facebook wall)
- Placement of user & application data



Worldwide Distribution

- Serve all users from the best datacenter (DC) with respect to **user perceived latency**
- Cloud service providers use many geographically dispersed DCs
- What data to store at which datacenter?
- **Interdependencies** between data items
- Minimize **operational cost** of the datacenters
  - Inter-DC traffic due to data sharing or interdependencies
  - Provisioned capacity at each DC

Replication

- Data replication for **fault-tolerance**
  - Hardware failures
  - Natural disasters
- Replication for **availability**
  - Large scale outages
- No single point of failure
- Replicas need to communicate frequently
  - Synchronization
  - Ensure consistency

Impacts of Data Placement

- **Latency** increases between distant locations
  - $\rightarrow$ Move data near the users that most frequently access it
- Amount of **inter-DC traffic** influences bandwidth costs
  - $\rightarrow$ Colocate data items
- **Capacity skew** among DCs increases hardware costs
  - $\rightarrow$ Uniformly distribute among DCs

Approaches to Data Placement

- How to find a good data placement that reduces latency and operational cost?
- **Full replication** at each datacenter
  - Lowest latency for the users
  - Excessive costs for DC operators
- **Single DC** holds all data
  - No inter-DC traffic
  - Many unhappy users due to high latency
- **Partition data** across multiple DCs
  - Challenging problem to find good placement
  - Need to analyze patterns of data access
  - Process $\gg 10^8$ objects

Challenges of Data Placement

- Cloud services deal with highly **dynamic data**
  - High update rates lead to stale replicas
  - Updates need to be visible worldwide
- Collaboration around the world
  - Users work together on a **shared data** item
- Data **interdependencies**
  - Publish-Subscribe mechanisms; “Friend of a friend”
  - Can be modeled as dependency graph
- Generate **huge data sets**
  - Need solutions for efficient analysis of the dependency graph

Challenges of Data Placement (cont'd)

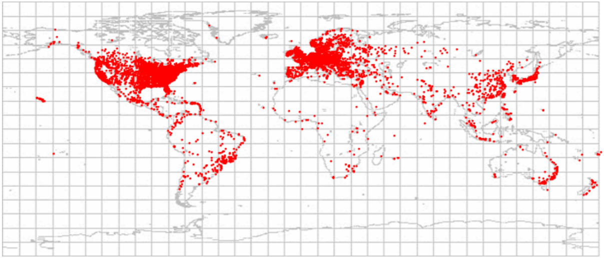
- Applications change frequently
  - Need to continuously adapt to **changing usage patterns**
- Increasing **user mobility**
  - When should data be migrated to new location?
- **Infrastructure** can change
  - Capacity limits or latencies between DCs may change

Network Traces

- Datacenter applications collect workload traces
- Month-long log from Live Mesh and Live Messenger
- Analysis focuses on the aspects of
  - Shared data
  - Data interdependencies
  - User mobility

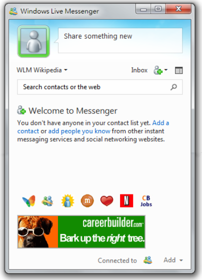
Live Mesh

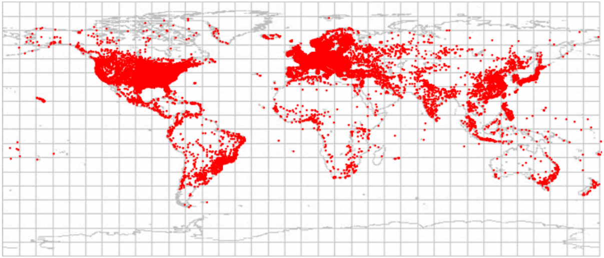
- File & Application synchronization
- Cloud storage
- Data feeds



Live Messenger

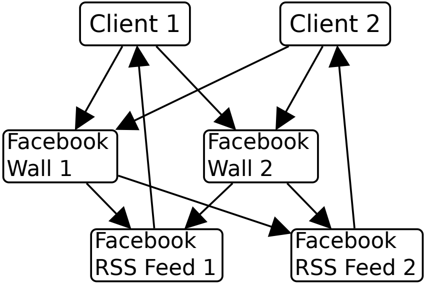
- Instant Messaging
- Video conferencing
- Continuous group conversation
- Contact status updates





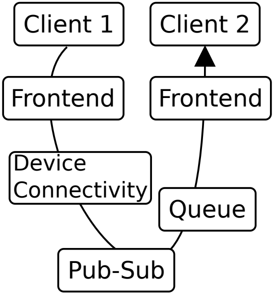
Facebook

- Facebook wall
  - Connects users to all of their friends
- Users can receive updates via RSS feeds
- Interdependencies between walls and RSS feeds



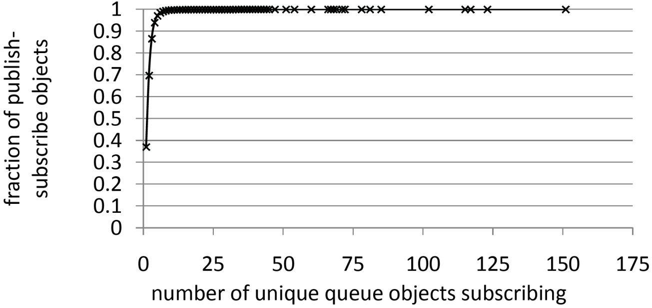
Data Sharing in Live Mesh

- Clients access Live Mesh through Web frontend
- Update to device connectivity status
- Multiple queue items can subscribe to publish-subscribe object



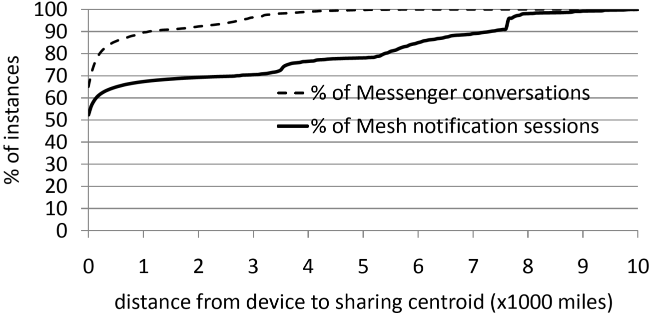
Data Interdependencies in Live Mesh

- Change to a document creates an update message at Publish-Subscribe objects
- Queue objects receive a copy of that message
- Long tail of very popular data items



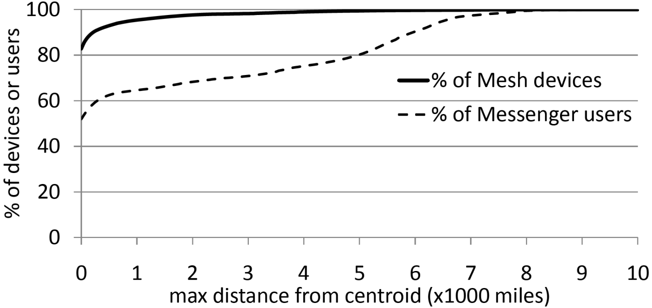
Geographically Distant Data Sharing

- Compute sharing centroid for each data item
    - Weighted mean between the users that access it
- Large amount of sharing occurs between distant clients



Client Mobility

- Geo-location database `quova.com`
  - Maps IP address to geographic region
- Centroid computed from all locations where the client contacted the service
- Large movements in the Live Messenger trace



Known Heuristics

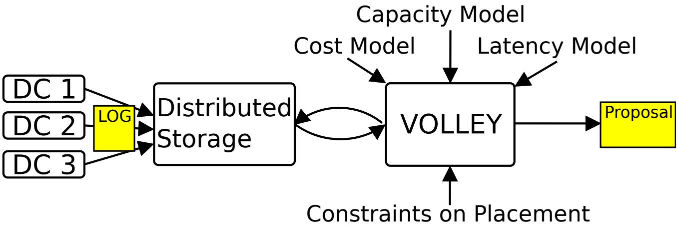
- Determine user location
- Move data to closest datacenter for that user
  - with the goal to reduce user latency
- Ignores major sources of operational costs
  - WAN bandwidth between DCs
  - Overprovisioned datacenter capacity due to skewed load

Volley's Approach

- Volley **optimizes** data placement for **latency**
  - and allows to limit operational **costs**
- Correlates **application logs** into graph that captures a **global view** on data accesses
- Analyzes data interdependencies and user behavior within cloud services
- Compute data placement and output **recommendations** when data should be **migrated**

Volley in a Nutshell

- **Input**: logs & models
  - Datacenter logs in distributed storage system
  - Models for cost, capacity and latency
  - Constraints on placement
- Iterative **optimization algorithm**
  - Distributed computing framework
- **Output**: migration recommendations



Requirements for Logs

- Capture logical flow of control across components

  → Construct dependency graph
- Provide unique identifiers for
  - data items: GUID
  - users: IP
- Request log record:
  - Timestamp
  - Source-entity: IP or GUID
  - Request size
  - Destination entity: GUID
  - Transaction ID: trace request in logs

Logged Events

- Live Mesh Trace
  - Changes to files
  - Device connectivity
- Live Messenger Trace
  - Login/Logoff events
  - Participants in each conversation
  - Number of messages between users

Datacenter Cost Model

- Cost per transaction, such as RAM, disk and CPU
- Capacity model for all DCs, e.g. amount of data stored at each DC
- Cost model for all DCs
- Models change on slower time scales

$\rightarrow$ Specify the hardware provisioning in DCs to run the service

$\rightarrow$ Required network bandwidth

$\rightarrow$ Charging model for service use of network bandwidth

Additional Inputs

- Location of each data item
- Model of latency
  - Network coordinate system: n-dimensional space specified by the model
  - Locations of nodes → predicted latency
- Constraints on placement
  - Replication at distant datacenters
  - Legal constraints

→ Allows to make placement decisions

Algorithm

Phase 1 Compute initial placement

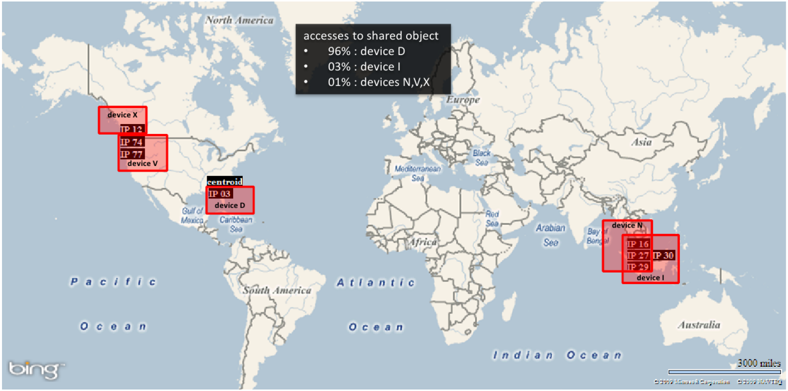
Phase 2 Iteratively move data to reduce latency

Phase 3 Collapse data to datacenters

Phase 1: Initial Placement

- Map data items to the weighted average of the geographic coordinates of the clients that access it
- Weight $=$ amount of communication client$\leftrightarrow$data item
- $\forall$ data items: compute **weighted spherical mean**
  - Interpolate between 2 initial points (clients)
  - Average in additional points
- Some data items may never be accessed directly by a client
  - Move them near the already fixed data items
- Ignores data interdependencies!

Phase 1: Initial Placement



Phase 2: Iteratively Improve Placement

- Move data items closer to users and other data items that frequently interact
- $\forall$ data items: determine movement to another node
  - Current latency and amount of communication increases the contracting force
- Updates to placement pull nodes together
  - Data items moveable
  - Client locations fixed
- Replicas treated as separate data items that interact frequently

$\rightarrow$ Reduce latency

$\rightarrow$ Reduce inter-DC traffic (if data items colocated)

Phase 3: Collapse Data to Datacenters

- Move data to nearest datacenter
- If DC over specified capacity
  - Identify data objects with fewest accesses
  - Move them to the next closest DC
- Iterations $\leq$ #DCs

Output: Migration Proposals

- Application-specific migration
- Supports diverse datacenter applications

- Proposal record:
  - Entity: GUID
  - New datacenter
  - Average latency change per request
  - Ongoing bandwidth change per day
  - One-time migration bandwidth

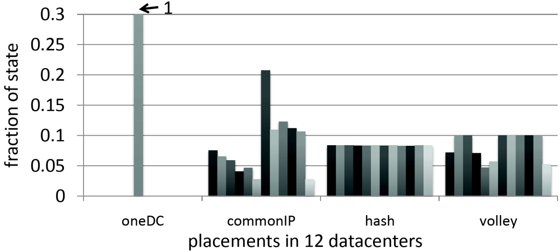
Test Environment

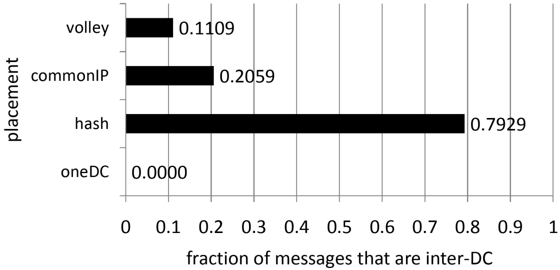
- Month-long Live Mesh trace
  - Compute placement on week 1
  - Evaluate placement on weeks 2-4
- 12 datacenters as potential locations
- Capacity limit: $\leq$ 10% of all data at each DC
- Analytic evaluation using network coordinate system

Heuristics

- commonIP
  - Place data near IP with most frequent access
  - Optimizes for latency
- oneDC
  - Place all data in one datacenter
  - Optimizes for zero inter-DC traffic
- hash
  - Place data according to hash function
  - Optimizes for zero capacity skew

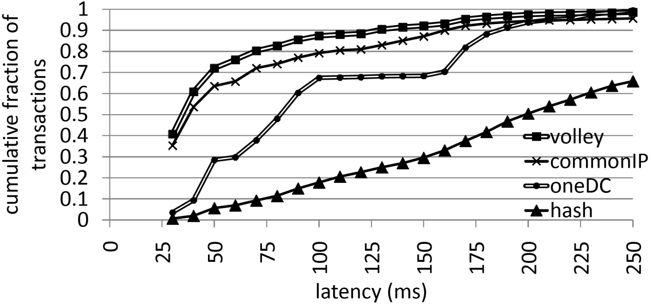
Capacity Skew & Inter-DC Traffic





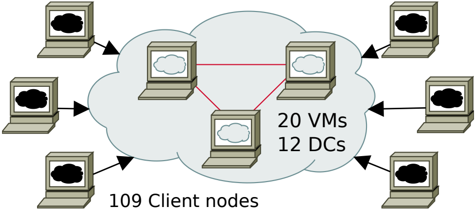
Latency

- Volley performs better than commonIP and provides
  - lower capacity skew
  - fewer inter-DC messages



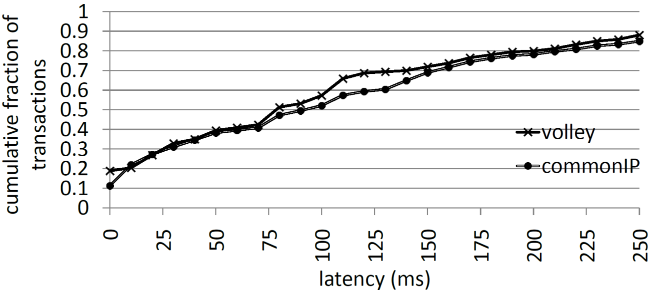
Live System Latency

- Live Mesh prototype
  - Frontend: allows clients to connect to any DC
  - Document Service: stores IP addresses of the clients
  - Publish-Subscribe Service: notifies about changes in the document service
  - Message Queue Service: buffers messages from the Publish-Subscribe Service
- Live Mesh trace replayed from 109 nodes scattered around the world



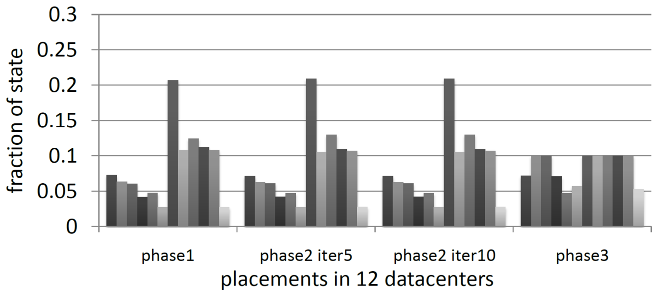
Live System Latency

- External sources of noise
    - Less client locations than real world scenario
    - Connectivity of the simulated clients does not conform to Volley's latency model



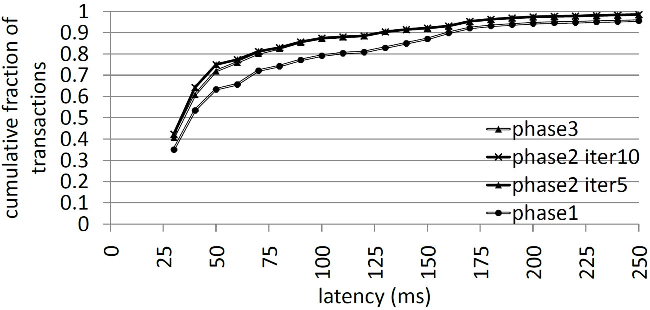
Impact of Iteration Count on Capacity Skew

- Most objects do not move after phase 1
- Capacity skew smoothed in phase 3



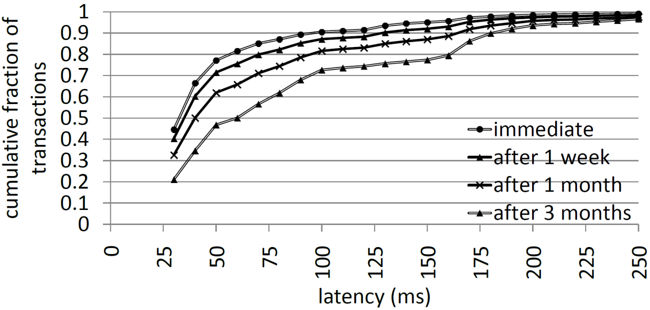
Impact of Iteration Count on Client Latency

- Latency remains stable after few iterations of phase 2
- Almost no penalty from phase 3



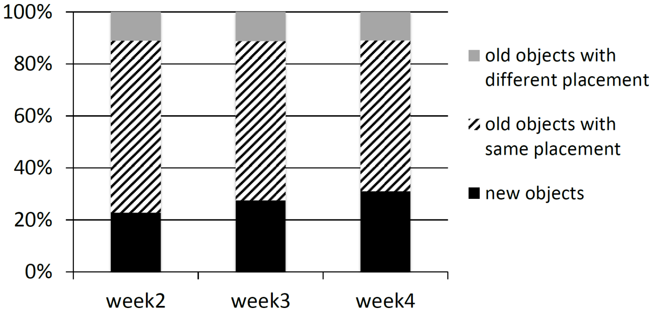
Re-Computation

- Volley should be re-run frequently
  - Stale placements increase request latency due to client mobility
  - Inter-datacenter traffic increases due to new objects that cannot be placed intelligently
  - Changing access patterns may require data movement
  - New clients need to be served



Migrated Objects

- Percentage of objects moved in placement computed after week X compared to first week
- Most old objects do not move



Summary

- Automatic recommendations for **data-placement under constraints**
- Placement can be controlled to
  - take resource usage into account (Cost & Capacity Models)
  - ensure replication (Constraints on Placement)
- **Application independence** allowing for specialized migration mechanisms
- Analysis of cloud services hightlighted the trends that motivated Volley: **shared data**, **data interdependencies** and **user mobility**
- Evaluation shows that Volley simultaneously **reduces latency** and **operational costs**
  - Improvement over state-of-the-art heuristic

Open Questions

- Volley handles placement decisions within cloud service
  - Extension to output **recommendations to DC operators** to upgrade their DCs or build new ones
- Can we allow new objects to be registered such that they get a good **initial placement**?
- Volley handles **replicas** as separate data items
  - Better alternative for modeling replicas?

Thanks for your attention