FogStore - Project Documentation

FogStore is a project of the Mobile Cloud Computing research group at Technische Universität Berlin and Einstein Center Digital Future.
The goal of the project is to build a distributed data management system for the fog to help application developers abstract from the complexities of fog computing.
This distributed data management system should support arbitrary, dynamically changeable data distribution policies as defined by applications while dealing with aspects such as QoS constraints or failures.
On the application side, a middleware component supports the application with data placement and replica selection decisions, which is especially useful for moving clients.
Finally, the project also encompasses a novel benchmarking framework to evaluate performance and other QoS properties of fog data management systems and other geo-distributed data management systems.

The project has been accepted and funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for a duration of three years as project 415899119.

Project Milestones

The FogStore project is split into four distinct work packages:

1. **Work Package 1** - Application-Controlled Replica Placement
2. **Work Package 2** - Dynamic Replica Discovery and Selection
3. **Work Package 3** - Predictive Replica Placement
4. **Work Package 4** - Fog Datastore Benchmarking and Evaluation

WP1 - Application-Controlled Replica Placement

FogStore starts with a *Core* work packages that serves as a basis for all future work.
It provides a fully working prototype that let's applications store data across different FogStore nodes, with FogStore handling all replication.
Applications can specify where data should be replicated using an API.

This work packages is split into the following deliverables:

WP1.1 - FReD (Fog Replicated Datastore) Software

The basic prototype for FogStore that lets application store data in keygroups in the fog.
Each node in the overall FogStore deployment can consist of one or more servers that each run the FReD service (WP1.1.1).
A client can identify any node using a unique logical address (WP1.1.2).
Each node has one storage backend where local replicas are stored: this can be either a local database or a cloud storage system (WP1.1.3).
Data is encapsulated into keygroups: each keygroup stores data items either in an append-only store or as a mutable table (WP1.1.4).
Replication control happens per keygroup, i.e. applications can set, at a keygroup-level, which node should have a replica (WP1.1.5).
A centralized store for the entire FogStore deployment exists that has knowledge about global replica placement and all keygroups and that acts as the single source of truth (WP1.1.6).
As an alternative to the centralized knowledge store, nodes in FogStore can manage replication knowledge using a consensus-based protocol and then propagate this information back to the central store - this should be configurable per FogStore deployment (WP1.1.7).
When a node is configured as a replica for a keygroup, an expiration for data items can be configured: data then expires on this replica node after a certain delay (WP1.1.8).
FogStore offers some basic access control mechanisms for multi-tenancy that allows applications to protect keygroups from read/writes from other applications (WP1.1.9).

- [ ] WP1.1.1 can be installed on single or multiple machines per node
- [ ] WP1.1.2 has logical addressing for geo-distributed sites
- [ ] WP1.1.3 use state-of-the-art or cloud storage system as storage backend (*Universal Storage Connector and H… Interface* (*USCHI*))
- [ ] WP1.1.4 keygroup stores data items in append-only store or mutable table
- [ ] WP1.1.5 applications choose replica placement
- [ ] WP1.1.6 centralized store has replica knowledge (*Holistic Application Naming Service* (*HANS*))
- [ ] WP1.1.7 alternative consensus-based replica set change
- [ ] WP1.1.8 data items can expire on a replica if needed
- [ ] WP1.1.9 access control mechanisms for multi-tenancy

WP1.2 - Trigger Node Software

Next to nodes that store replicated data, a FogStore deployment also comprises *trigger nodes* using *Keygroup Update Replication Triggers* (*KURT*).
A trigger node can be configured at a keygroup level.
Trigger nodes receive all updates on data items on a specific keygroup.
A trigger node may then use this data for external systems or to write modified data back into FogStore.

WP1.3 - Evaluation: Central Replica Store vs Distributed Consensus

WP1.1.6 and WP1.1.7 describe two mechanisms to manage consistency of information about keygroup replication.
Those two mechanisms should be compared experimentally.

WP2 - Dynamic Replica Discovery and Selection

The goal of this work package is to enable application clients, especially moving ones, to always connect to an optimal FogStore node.
This encompasses discovering nodes and selecting the optimal node given criteria such as (i) if needed data is replicated to that node, (ii) bandwidth and latency constraints, and (iii) node performance.

This work packages is split into the following deliverables:

WP2.1 - Application-Side Middleware

The *Application Level Extension to Allow Node Discovery and Replica Appointment* (*ALExANDRA*) is a middleware for applications through which a connection to FogStore can be abstracted.
This middleware connects to FogStore nodes running the FRed service and manages authentication, and passes data CRUD requests and updates to replication configuration (WP2.1.1).
This application extension is also responsible for finding the optimal replica node and connects to this node automatically and without downtime for the application (WP2.1.2).
To this end, the middleware collects metrics on availability of and communication latency to possible candidates in a lightweight monitoring module (WP2.1.3).
We also explore leveraging this middleware to improve consistency guarantees for application clients (WP2.1.4).

- [ ] WP2.1.1 connects to FReD nodes and passes requests from applications
- [ ] WP2.1.2 connects to logically closest replica (discover + select)
- [ ] WP2.1.3 collects metrics in lightweight monitoring module (availability + communication latency)
- [ ] WP2.1.4 improves consistency guarantees for clients ("A Middleware Guaranteeing Client-Centric Consistency, D. Bermbach, 2013)

WP2.2 - Extended FReD Software for Replica Discovery and Selection

To enable WP2.2, the ALExANDRA middleware requires additional APIs from FogStore.
The FReD software presents the necessary information on network location, availability, replication, etc. to the middleware over such APIs (WP2.1.1.).
FogStore nodes also communicate with other nodes in close proximity to discover replicas of hitherto unknown keygroups, using the HANS (WP1.1.6) only as a fallback.
To this end, a gossip-based search protocol using message piggybacking is leveraged, including a message hop counter to drop messages after too many hops.
This information benefits the ALExANDRA middleware as it can receive information about nearby nodes (WP2.2.2).
Similar to ALExANDRA, FReD also contains a lightweight monitoring module to collect metrics on availability of and communication latency to nearby nodes in the FogStore deployment (WP2.2.3).

- [ ] WP2.2.1 present necessary information to facilitate discovery and selection process in application middleware
- [ ] WP2.2.2 gossip-based search protocol to find replicas of unknown keygroups based on geographic proximity (use naming service only as fallback, exploit message piggybacking, message hop counter to drop messages)
- [ ] WP2.2.3 lightweight monitoring module to collect information about latency and availability of other nodes (similar to app middleware)

WP2.3 - Continuously Evolving Logical Distance Graph

The information on availability and communication latency collected by monitoring modules in ALExANDRA (WP2.1.3) and FReD (WP2.2.3) is used to create a continuously evolving logical distance graph that has the distance between all FogStore nodes and that can be used by ALExANDRA to select the closest replica that is likely to be available for an application.
This graph needs to be shared to all nodes and, to balance the tradeoff between information completeness and necessary memory footprint of this distance graph, is CRDT based.
Locally, standard graph theory algorithms may then be used to select closest replicas.

WP3 Predictive Replica Placement

The main goal in this work packages is to proactively provision replicas in FogStore by anticipating future application requests.

WP3.1 - Location Selection

The first part of proactive provisioning is predicting future locations of application clients.
To this end, the ALExANDRA middleware predicts those locations using historical application request distributions and *application hints*, where applications can use an API exposed to them by ALExANDRA to inform of future client movement.
ALExANDRA then uses this information to identify logically close nodes that have sufficient capacity to replicate the needed data.
In addition to predicting future locations, the middleware also predicts unlikely future locations to remove nodes from the replica set.

WP3.2 - Placement Decision

The second part of proactive provisioning is using predicted future locations to decide where to place keygroup replicas.
A new replica set is formed using a combination of existing replicas and a suitable subset of candidate nodes as identified by the application-side middleware.
Here, data movement restrictions and limits to the number of replicas for a keygroup are complied with.
This decision process considers the cost of moving replicas but also opportunity costs of not moving replicas.
It runs in ALExANDRA, FReD, or a combination of both.
We also allow temporary partial replicas of data to increase performance.

WP3.3 - Analyze Different Sources of Information and Prediction Methods

Application prediction can use different sources of information (WP3.1) and different prediction methods.
We analyze prediction accuracy between using historical application request distribution, using applications hints, and using a combination of both.
We also explore using different prediction methods such as machine learning and ARIMAX time-series analysis.

WP4 - Fog Datastore Benchmarking and Evaluation

As a final part, we develop a fog benchmarking tool and evaluate FogStore.
This is split into four parts:

WP4.1 - New Metrics and Measurement Approach for Fog Application Benchmarks

First, we determine metrics for fog application benchmarks.
Next to throughput, we must also consider data consistency and availability of the fog system.
Second, we develop a measurement approach.
This measurement approach should reflect real-world fog applications and, subsequently, consider different workloads

1. stream-based continuous data inflow
2. push-based outflow
3. traditional OLTP

- should provide insights into data consistency and performance
- consider geo-distribution of system and clients

WP4.2 - Fog Application Workloads

- different workloads at the same time
- analyze real-world fog applications + general geo-distributed systems

WP4.3 - Fog Application Benchmarking Tool

- coordinate geo-distributed, multi-workload benchmarks
- no negative impact on reproducibility and accuracy
- prefer retrofitting existing tools, fallback to completely new tool

Geo-Replicated benchmarkinG (GeoRG)

WP4.4 - Evaluate FogStore

- benchmark different features using new benchmarking tool
- analyze applicability of FogStore to real applications (external partners + student projects)