The Anzere personal storage system

Oriana Riva Qin Yin* Dejan Juric Ercan Ucan* Robert Grandl Timothy Roscoe

*Systems Group, Department of Computer Science, ETH Zurich (*Student)*
{oriva,qyin,eucan,troscoe}@inf.ethz.ch {djuric,rgrandl}@student.ethz.ch

1 Introduction

As an alternative to commercial online service providers, recent projects (Cimbiosys, Perspective) have proposed systems that replicate a user's personal data (photos, music, etc.) across his or her devices based on user-specified policies. Supported policies are usually limited to device-specific selection predicates. For example, a phone might have a filter that selects photos tagged 5-star rating. This kind of approach not only limits the expressivity of policies, but also binds the replica to specific devices. Instead, we would like policies to be independent of the personal ensemble for which they are initially specified, and thus continue to work despite devices that leave, break, or get replaced.

In the *Anzere* project, we are exploring a new replication approach which dramatically increases the expressivity of the policy language, but still scales policy calculations to large numbers of items. This approach is implemented in the Anzere runtime, currently running in personal ensembles of mobile phones, laptops, desktop PCs and rented cloud resources (from Amazon EC2).

2 Anzere model

In Anzere, policies are not only content-neutral, but also *device-neutral*. Rather than referring to devices and content by names, we use predicates. For example, to provide durability of videos taken with the phone, rather than a policy like "Replicate videos on my phone to my home server", a user can specify "Ensure one copy of every video exists on a device I own". Specifically, policies are triplets `<IP,DP,R>`, comprising an item predicate `IP`, a device predicate `DP`, and a relationship `R` that must hold between the items and devices selected by the predicates. The above policy is represented as: `policy([[video_item]],[repany],[[owned_device]])`.

With this approach, Anzere allows users to compose a large range of policies, from accessibility ("Replicate recently-downloaded music on a device I carry") and durability ("Keep at least 3 distributed copies of my thesis") to privacy ("Do not store private data in the cloud") and capacity limits ("Allow to 2GB free on the phone").

We express policies using a logic programming language – Prolog in our implementation. Logical unification is particularly useful to represent information extracted from a set of heterogeneous sources (as in our personal ensemble), without binding the system to a predefined schema. To evaluate policies we apply *constraint logic programming* (CLP). We use constraints to express policy requirements such as number of replicas, resource limitations, or rental fee for cloud storage. A CLP solver then computes the best data placement across the personal overlay which respects the given constraints. In doing this, rather than reasoning item by item, which would lead to an unacceptable solving time, items are grouped in *equivalence classes*, which are derived directly from the `IPs` in the policy set. For example, from the policy above, two equivalence classes follow: `[video_item]` and `[¬video_item]`. Experiments have shown equivalence classes allow Anzere to scale to very large numbers of items and complex policies.

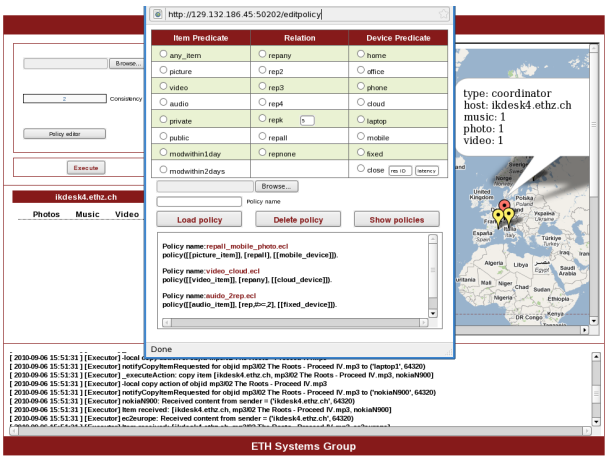

Figure 1: Anzere user interface.

3 Implementation and demo

The architecture of Anzere include overlay network, monitoring infrastructure, CLP solver, knowledge base, data replication, and Paxos. We will show a demo of Anzere running on laptops, Nokia N900 phones, PCs, and cloud VMs. Through the UI (in Figure 1), users can control the personal storage overlay, compose policies, submit media files, and monitor the solver's execution.