FSaaS:
Configuring Policies for Managing Shared Files Among Cooperating, Distributed Applications

Marco Di Sano, Department of Electrical, Electronic and Computer Engineering, University of Catania, Catania, Italy

Antonella Di Stefano, Department of Electrical, Electronic and Computer Engineering, University of Catania, Catania, Italy

Giovanni Morana, Department of Electrical, Electronic and Computer Engineering, University of Catania, Catania, Italy

Daniele Zito, Department of Electrical, Electronic and Computer Engineering, University of Catania, Catania, Italy

ABSTRACT

In this paper, the authors introduce and describe the concept of File System as a Services (FSaaS), an highly configurable cloud service that enables cooperating, distributed applications to define their own rules and policies for managing sets of files shared. The FSaaS aims to create a logical virtual space, containing references to shared files, whose management layer supports the same functionalities of a file system (basic file operations) but where each single file can have different policies for consistency, synchronization and replication. This work explains the idea at the base of FSaaS, describes in details its main components and their interactions and illustrates two use cases for better explaining the provided functionalities.

Keywords: *Cloud Computing, Concurrency, Consistency, Distributed File System, File Sharing, Logical Virtual Space*

INTRODUCTION

Cloud Computing (Armbrust et al., 2009; Voorsluys, Broberg, & Buyya, 2011; Beloglazov et al., 2011) is a distributed computing paradigm on which both hardware (computation, storage and network) and software (OSs, databases, Web servers, as well as scientific, office and CAD suites) resources are made available, on demand, from anywhere through the Internet. Today, many services providers use one of the different cloud services models (IaaS, PaaS, SaaS) to build highly configurable, scalable and reliable environments where host and execute their applications.

The operation of most applications on cloud infrastructures is based on the exchange of common data and information (Di Stefano,

DOI: 10.4018/jwp.2013010101

Morana, & Zito, 2012), through shared files. There are several solutions for sharing files in distributed environments (Coulouris at al., 2012), ranging from the simpler file hosting services, such as Dropbox (2012), to the more complex Distributed File Systems (DFS) (e.g. NFS (Sandberg, 1986), AFS (Howard et al., 1988), GFS (Ghemawat et al., 2003), Hadoop (Shvachko et al., 2010), Cassandra (Lakshman & Malik, 2009), Dynamo (De Candia et al., 2007), HekaFS (Darcy, 2012)). However all the present solutions have featured by fixed management policies with precise semantics to control the access, to handling concurrency, to maintain consistency and replicas. All these solutions propose a well-defined functional scheme, poorly configurable, which may be suitable for some classes of applications but it is inadequate for others.

The semantic options for files management are completely established by the native file system and they cannot be configured and adapted by the application designer. The operations on a file are restricted to the supported features of the chosen DFS.

Many cloud scenarios are highly dynamics in that number and type of shared files, as well as their management policies, could vary frequently, thus they could obtain advantages by adopting applications– dependent file sharing strategies.

This paper, an extended version of Di Sano et al. (2012), introduces the concept of *File System as a Service* (FSaaS), proposing a new approach to overcome the lack of configurable file sharing systems. In particular, a configurable file manager allows a group of applications to specify, for each shared file, the behavior and the semantic of each file operation (i.e., open, read, write) in order to satisfy the requirements for consistency, synchronization and replication. This characteristic represents the main contribution of the present work. It allows users to define multiple management policies for different applicative scenarios.

The rest of the paper is structured as follows. First we give an overview of related works proposed in literature. Then we introduce the concept of FSaaS and illustrates some design considerations. A description of the FSaaS architecture is provided afterwards. Followed by an illustration of two use cases. Finally, we conclude the work.

RELATED WORK

There are a lot of studies dealing with the file sharing in distributed systems.

Network File System (NFS) (Sandberg, 1986) is a way to share files among machines on a network as if they were located in the client's local hard drive. The main idea of NFS is that each file server provides a standard view of its file system. The NFS protocol allows clients (which are heterogeneous processes) to access files on a remote server, in order to share a common file system. NFS uses a remote access model where the client asks to the file server (through a common API) for executing some operations on a set of files. It is different from the upload/download model, where the client first downloads the affected file, then modifies and uploads it on file server. The main advantages of NFS are transparency on access and good failure robustness, but there is no migration transparency. If a resource is placed in other server, the client must be aware of this change. Moreover, NFS has a very low availability and poor scalability since it is based on a single server model, even if there are multiple servers (each one runs independently).

Andrew File System (AFS) (Howard et al., 1988) was born with the main goal to optimize scalability, in order to achieve better performance even in the presence of a large number of client applications. However, AFS makes some assumptions and creates its infrastructure and communication protocols based on the following considerations: most files are small, reads are much more common than writes, most files are read/written by one user, files are referenced in burst (locality principle, so once referred, a file will probably be referenced again). AFS uses Kerberos for authentication and implements access control list (ACL) on

Squiride Rank: Squirrel Ride Rank Algorithm-Based Feature Extraction for Re-Ranking of Web Pages
Lata Jaywant Sankpaland Suhas H. Patil (2022). *International Journal of Web Portals (pp. 1-23).*
www.irma-international.org/article/squiride-rank/298990