Arquitectura en capas para acceso remoto SAD

Karina Cenci – Leonardo de - Matteis – Jorge Ardenghi

Laboratorio de Investigación en Sistemas Distribuidos
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
{kmc,ldm,jra}@cs.uns.edu.ar

Resumen La utilización de fuentes de datos compartidas facilita el trabajo en equipo. Por este motivo las organizaciones tienen implementan en sus locaciones sistemas con controles de acceso para compartir los datos de acuerdo a los privilegios de los usuarios. Nuevas formas de trabajo son la distribución de los miembros de un equipo en distintos lugares físicos, el trabajo desde las casas de los empleados, el traslado temporal a otra locación. Por todas estas razones, acceder a los datos en forma remota es una necesidad en crecimiento. En tal sentido, un punto a tener en cuenta es el costo de los recursos de transporte necesarios para generar la comunicación. Una respuesta a esta necesidad es la propuesta de una arquitectura referente en capas ICSAD (Interfaz, Control y Sistemas de Archivos Distribuidos). La misma permite construir una implementación que facilita la descarga de los documentos y el control de versionado para el caso en el que varios usuarios estén accediendo en modo modificación.

Palabras claves: Sistemas de Archivos Distribuidos - Acceso Remoto - Sistemas Distribuidos

1. Introducción

La utilización de recursos desde distintos espacios geográficos se ha incrementado y expandido a partir del auge de los dispositivos móviles, las redes de banda ancha y las conexiones inalámbricas. Las nuevas tecnologías motivan a las organizaciones a solicitar nuevos requerimientos para mejorar la calidad y eficiencia del trabajo de sus empleados.

El acceso a recursos remotos, en especial a fuentes de datos, ha motivado una variedad de estudios [8], [4], [6], [7]. Weissman y otros [8] proponen un paradigma denominado *Smart File Object* (SFO) para alcanzar un rendimiento óptimo en el acceso a archivos remotos. En el mismo, utilizan el concepto de archivo como un tipo de objeto para proveer una interfaz de alto nivel y hacer uso de los conceptos de objetos para la invocación de las operaciones y propiedades de los archivos. El sistema de archivos *Trellis* [7], por su parte, provee una abstracción para acceder a archivos de datos utilizando nombres de archivos basados en *URL* y *SCL* y sus funcionalidades básicas están implementadas en el espacio de usuario. Una de las características que presenta es el acceso transparente a cualquier dato remoto, una capacidad importante para las áreas de metacomputación y *grid computing*.

Por otra lado, la administración de datos compartidos dentro de una empresa u organización, comúnmente se realiza a través de un sistema de archivos distribuidos o de red, con capacidades para el manejo de usuarios, permisos en una red de área local o nube propia. En algunos casos, estos sistemas no ofrecen seguridad, escalabilidad y operabilidad traspasando los límites organizacionales.

La alternativa de utilización de una red privada virtual (*VPN*) [3], en algunos casos, no es una vía de solución aceptable para las empresas, ya que no respeta las políticas de seguridad informática establecidas internamente. Cabe destacar, en este punto, que no es recomendable brindar a todos los usuarios acceso *VPN* a la red local de una organización por motivos varios de seguridad. Además, para la implementación de este tipo de accesos deben considerarse factores de rendimiento como la velocidad del enlace disponible y utilización de CPU (tanto en el cliente como en el servidor de *VPN*) según el esquema de cifrado del protocolo adoptado (PPTP, L2TP/IPSec, OpenVPN, etc.).

Miltchev y otros [6] establecen un *framework* para comparar diferentes sistemas de archivos distribuidos. En el *framework* identifican las características necesarias para esta comparación: autenticación, autorización, granularidad, delegación autónoma y revocación. Estos criterios de comparación permiten alcanzar un entendimiento de las soluciones intermedias para el acceso a datos compartidos. La relevancia de los tópicos seleccionados para la comparación está en que el acceso remoto requiere de manejo de credenciales, autorización, permisos para habilitar o no el acceso a los datos, en este caso, el acceso a los archivos.

A continuación se detallarán brevemente algunas propuestas para acceder a archivos compartidos existentes y de uso común en algunas organizaciones. Luego, más adelante, propondremos una arquitectura que pretende cubrir mayor cantidad de aspectos y funcionalidades necesarias hoy en día para el acceso a archivos compartidos desde fuera de la red interna de una organización. Para esto último presentaremos un ejemplo concreto de aplicación y detallaremos los componentes de la implementación. Por último, al final del presente trabajo se expondrán ventajas y desventajas de la propuesta presentada junto con las conclusiones del caso y desarrollos futuros.

2. Otros antecedentes

Entre las alternativas existentes para gestionar el acceso a archivos compartidos y de uso habitual en las organizaciones actuales pueden mencionarse *HFS*, mod_dir de Apache y WebDAV.

En primer lugar, *HFS* (HTTP File Server) [5] permite compartir archivos fácilmente entre un grupo de trabajo a través del protocolo *HTTP*. La compartición de los archivos se puede limitar a un grupo de usuarios o permitir que todos puedan acceder a los mismos. La diferencia con respecto a otros sistemas de archivos es que no se requiere de una red. *HFS* es un servidor web, esta característica habilita a que se publiquen los archivos a través de una presentación de un website. La utilización del protocolo *HTTP* presenta debilidades en el aspecto de seguridad, ya que el tráfico es transmitido en texto plano y cada bit de dato transportado entre el servidor web y el cliente puede ser interceptado y leído por todos los equipos que están en la cadena que pasa los datos al destino final. Esta herramienta funciona sobre el sistema operativo Microsoft Windows. Una desventaja que presenta es que no provee un esquema de autenticación a través de la interface *ADSI* (Active Directory Services Interfaces).

Una segunda alternativa, *Apache Module* mod_dir [1] se utiliza para redireccionar barra final (*trailing slash*) y servir como índice de directorio de archivos. Es una forma

simple que se utiliza para compartir archivos, en especial es usada en las fuentes de datos de libre distribución para acceder a los servidores. Una de las ventajas de esta herramienta es que se puede instalar sobre diferentes sistemas operativos sobre los cuales se puede ejecutar Apache HTTP Server como: Microsoft Windows, GNU/Linux, Unix, OS X, etc.

En tercer lugar *WebDAV* (Web-based Distributed Authoring and Versioning) es un conjunto de extensiones de HTTP, que permite a los usuarios colaborar entre ellos para editar y manejar archivos en servidores web a través de la red. *WebDAV* está documentado en RFC 2518 y extendido en RFC 3253, RFC 2518 especifica el conjunto de métodos, encabezados y tipos de contenido secundario a HTTP/1.1 para el manejo de propiedades, creación y administración de colecciones de recursos. Para usuarios comunes, *WebDAV* permite a equipos de desarrollo web y otros grupos de trabajo utilizar un servidor web remoto tan fácilmente como un servidor de archivos local.

Un ejemplo de utilización de *WebDAV* es presentado por Hernández y Pegah [2]. Para que el acceso compartido a *WebDAV* sea continuo (sin interrupciones) se le incorpora *LDAP* (Lightweight Directory Access Protocol) en el sistema para mantener una única registración. Todo esto se integra a través del servidor web *Apache* que permite la utilización de las extensiones *WebDAV* y el modelo de meta directorio *LDAP* para la autenticación de usuarios. La ventaja de esta implementación es que ofrece una solución compatible con *NFS* y OS X.

3. Arquitectura referente - ICSAD

El desafío al que intenta dar una respuesta la propuesta que presentamos en este trabajo es brindar acceso remoto desde distintos tipo de dispositivos, garantizando que se respeten las políticas de seguridad utilizadas dentro de los límites de la organización. Así la arquitectura referente, que denominamos Interfaz, Control y Sistema de Archivos Distribuidos (ICSAD), se muestra en la figura 1.

Como puede apreciarse, el modelo propuesto está organizado por capas. Cada capa es independiente, y se comunican a través de las interfaces definidas, de tal manera que si se modifica el comportamiento de las funciones no sea necesario modificar el resto de los componentes.

Los componentes principales son el sistema de archivos distribuidos (SAD), el módulo de control y el módulo de interfaz.

- Sistema de Archivos Distribuidos (SAD): este componente se encuentra dentro de los límites de la organización. Incluye todas las operaciones para el manejo de los archivos, las capacidades de acceso, la compartición de los directorios y archivos, la administración de los usuarios y permisos.
- Módulo de control: este componente es el encargado de conectar al módulo de interfaz con el sistema de archivos distribuidos, es la puerta de entrada a la organización desde el exterior. Incluye funciones para garantizar la seguridad en el acceso, permitiendo a los usuarios acceder a la información permitida desde el componente SAD. Además, se incluyen las funciones para leer, copiar, modificar, agregar documentos en el SAD.
- Módulo de interfaz: este componente se ejecuta en cada uno de los puntos de acceso remoto, como puede ser un teléfono celular, *tablet*, *notebook*, etc. Todas las operaciones requeridas sobre el SAD se realizan a través del módulo de control.

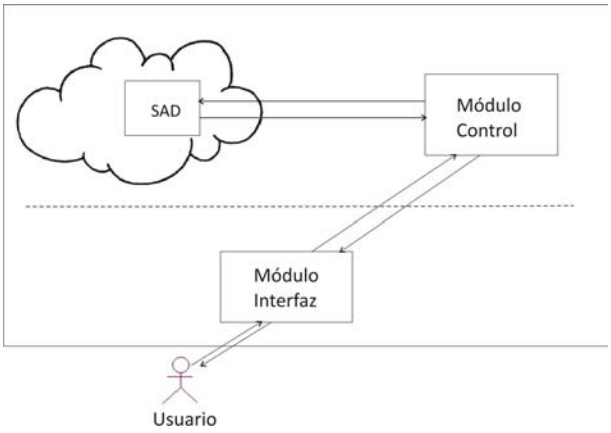

Figura 1. Arquitectura referente del Modelo ICSAD

El componente principal de esta arquitectura es el módulo de control. Que incluye las siguientes funciones:

- Lectura: la política utilizada para la implementación de esta operación es la descarga del archivo (*downloading*).
- Escritura: esta operación es equivalente a la creación de un nuevo documento en el sistema de archivos, considerando que no puede tener el mismo nombre que un archivo existente.
- Modificación: en este caso se trata de modificar un archivo existente, para ello se puede adoptar una de las siguientes políticas en el módulo de control:
  - Semántica de sesión: cuando un archivo es modificado por varios usuarios en el mismo instante de tiempo se guarda en el sistema de archivos la última copia cargada (*uploading*).
  - Semántica de versionado: en este caso, se almacenan todas las versiones del archivo bajo el mismo nombre pero con algún atributo distintivo, como puede ser el usuario, la fecha y hora de carga del archivo, o bien un identificador interno.

4. Ejemplo de aplicación

Para la arquitectura propuesta se diseña una implementación de un esquema de acceso de remoto a archivos en un repositorio distribuido ubicado sobre la red interna de una organización.

Los requisitos para la implementación que se consideraron fueron los siguientes:

- Tener acceso remoto a los archivos comunes de la organización.
- Respetar los mismos permisos de acceso sobre carpetas y archivos que brinda el SAD.
- La principal funcionalidad es el acceso en modo lectura a los archivos.
- No se podrán borrar carpetas ni archivos.

- Como funcionalidad secundaria es el acceso de escritura sobre archivos.

En el primer caso, en los repositorios se ubican los archivos compartidos, que son accesibles actualmente vía protocolo SMB (*Server Message Block*) sobre los servicios de Microsoft AD (Active Directory) de Microsoft Windows 2003R2.

Los usuarios de diferentes áreas de trabajo pueden leer aquellas carpetas y archivos sobre los que tienen permisos explícitos de acceso, otorgados según políticas de organización de la empresa. La funcionalidad principal es brindarle a los usuarios un servicio de acceso remoto a los documento internos, utilizando las mismas medidas de seguridad y políticas de acceso como si estuvieran en sus estaciones de trabajo en la red local.

Las políticas seleccionadas en el módulo de control son las siguientes: 1) para la lectura, la descarga del archivo al dispositivo; 2) para la escritura, no se permite crear un documento con el mismo nombre de otro documento en la carpeta correspondiente; 3) para la modificación, se optó por la semántica del versionado. En la selección entre las alternativas posibles para la función de modificación se consideró importante la posibilidad de brindar información a los usuarios de todas las modificaciones realizadas en un período de tiempo concurrente.

4.1. Componentes

Para la implementación de un entorno de servicio que cumpla con los requisitos funcionales especificados en la sección anterior se utilizaron los siguientes componentes:

- Máquina virtual con sistema operativo GNU/Linux distribución CentOS 6.
- Componente smbclient del producto Samba.
- Un servidor web Nginx.
- Utilización PHP-FPM para interactuar con el servidor web.
- Servicio de scripting a través del módulo PHP-CLI junto con el núcleo de PHP en su versión 5.4
- Módulo $ngx_http_ssl_module$ para proveer implementar soporte HTTPS.
- Librería OpenSSL.
- Módulo de control implementado en lenguajge PHP.
- Interface web implementada en lenguaje PHP.
- Dispositivos móviles con navegadores web.

La selección de los componentes estuvo guiada por las ventajas que ofrecen para el propósito del entorno formulado. Así, PHP es un lenguaje de *scripting*, hoy en día catalogado como de propósito general, de ejecución en servidores web. En la implementación los módulos desarrollados en PHP se ejecutan a través una interface FPM que permite un mayor rendimiento y mejores velocidades en relación a las que se hubieran alcanzado eligiendo un despliegue que utilizara el servidor web Apache y PHP cargado como un módulo del mismo.

Por su parte, PHP-FPM (FastCGI Process Manager) es una interface *FastCGI* siendo una implementación alternativa con características adicionales que hacen apropiado su uso en aplicaciones web de cualquier tamaño pero con alta cantidad solicitudes por unidad de tiempo.

En tal sentido, *FastCGI* es un protocolo que hace de interface con programas que se comunican con un servidor web. Es una variación del método de comunicación denominado *CGI* (Common Gateway Interface). El objetivo principal de este protocolo

es disminuir los tiempos adicionales asociados a la comunicación con un servidor web determinado, permitiendo al servidor web atender mayor cantidad de requerimientos al mismo tiempo.

Para proveer comunicaciones seguras entre el módulo de interfaz y el módulo de control, se optó por implementar un esquema de comunicación basado en el protocolo *HTTPS*, de esta manera se alcanzan los objetivos referentes a las políticas de confidencialidad de los datos de la organización. Además se utilizan directivas *HSTS* (HTTP Strict Transport Security) para el acceso seguro al módulo de interfaz.

La figura 2 muestra un diagrama lógico con los componentes enunciados para facilitar la comprensión del ejemplo de aplicación propuesto. En ella, la pila VM representa al módulo de Control, Clientes al módulo de Interfaz y AD al SAD conteniendo el repositorio de datos compartidos.

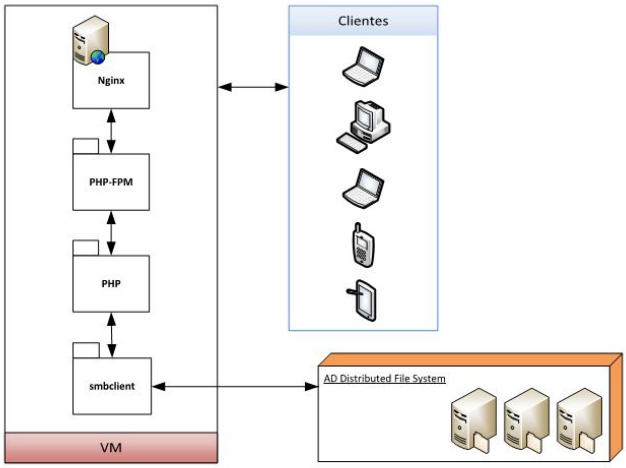

Figura 2. Diagrama de componentes de una implementación posible.

5. Conclusiones

La inserción de los medios de comunicación en la sociedad modifica la forma de efectuar las actividades cotidianas por parte de los usuarios, por ejemplo, el uso de Internet para realizar pagos, consultas, reservas, servicios de gobierno, etc. En el caso de las empresas, la conectividad a la red permite que los empleados puedan llevar a cabo sus actividades laborales desde distintas locaciones. Una alternativa es la utilización de *VPN*, que presenta ventajas y desventajas.

Como alternativa, en este trabajo se propone una arquitectura referente ICSAD para modelar el acceso a fuentes de datos que se encuentran dentro de los límites

de la organización. Esta alternativa garantiza que cada usuario obtenga las mismas capacidades y permisos como si estuviera trabajando dentro de los límites físicos de la organización.

En la arquitectura propuesta podemos definir la autenticación sobre diferentes implementaciones de archivos distribuidos que hagan uso de *LDAP* o bien algún otro tipo de sistema para catalogar usuarios y objetos con diferentes permisos, ya que el módulo de control es el encargado de la autenticación del usuario remoto.

La autorización se deja al sistema subyacente que provee dicho servicio. Así, tanto si es necesario hacer una validación sobre cada requerimiento —como en el ejemplo de implementación presentado— como si se obtiene una validación perdurable por un tiempo determinado —por ejemplo, en el caso del empleo de la API ASDI— el módulo de control puede adaptarse para ambos esquemas.

Otra ventaja de este esquema de validación de permisos es que, si el administrador modifica permisos para los objetos del sistema de archivos distribuidos, estos inmediatamente van producir un efecto en los archivos que el usuario puede o no acceder.

Por otra parte, la arquitectura provee acceso a través de la interface gráfica que se le presenta al usuario sobre cualquier dispositivo móvil mediante un navegador con soporte de protocolo *HTTPS* y lenguaje *HTML*. El objetivo es brindar un acceso universal a todos los equipos clientes sin necesidad de instalar una aplicación compilada específicamente para cada arquitectura y sistema operativo disponible hoy en día en dispositivos de uso masivo. Los usuarios no se crean sobre esta arquitectura: ya existen sobre el esquema de archivos compartidos que tenga la organización.

Además, de esta manera, se aseguran requisitos de seguridad básicos: autenticación, privacidad y auditoria. Con respecto a los dos primeros se deducen del detalle presentado en párrafos anteriores, pero podemos comentar que el tercer requisito, la auditoria, es una ventaja de la arquitectura, ya que los administradores de la organización podrán contar con un registro de todos los sucesos sobre el acceso a los archivos internos de la empresa desde el exterior. Se podrán contabilizar las autenticaciones, las descargas y modificaciones hechas por cada usuario. Cabe destacar que sin un módulo de control que provea estos servicios, en los trabajos relacionados presentados anteriormente no se dispone de toda la información necesaria para una auditoria completa, salvo que el administrador del sistema modifique compartimientos propios de dichas propuestas para incluir este tipo de auditorias, bien modificando políticas propias del sistema de archivos distribuidos (por ejemplo, en el caso de implementaciones sobre Microsoft AD) o bien *logs* de mod_dir en Apache, por citar algunos casos particulares.

Otra característica ventajosa es el hecho de que no se expone la red interna complementamente a los equipos remotos que utilizan los usuarios para acceder, como ocurre en el caso del empleo de redes privadas virtuales (VPN). Esta propuesta permite que en organizaciones que no implementan VPNs como un servicio para todos los usuarios, por sus políticas internas, se brinde este servicio en forma más general sin los inconvenientes de seguridad que presentan las VPNs. Es decir, sólo aquellos usuarios con excepciones y privilegios especiales podrán hacer uso del servicio de VPN (si estuvieran implementadas en la organización).

Como proyeccciones futuras se plantea la incorporación al módulo de control de submódulos para acceso a diferentes sistemas de archivos distribuidos, junto con la característica de autenticación sobre diversos sistemas que tengan LDAP como un servicio asociado, o bien otros esquemas de validación posibles. Por consiguiente, se aspira a poder especificar una estructura general para el módulo de control como objetivo final, para que éste, a su vez, pueda incorporar otros submódulos en la medida que resulta

necesario, a fin de ampliar el espectro de uso de la arquitectura de referencia planteada en este trabajo.

Referencias

1. The Apache Software Foundation. *Apache Module mod_dir*, 2013. httpd.apache.org/docs/current/mod/mod_dir.html.
2. L. Hernández and M. Pegah. Webdav: What it is, what it does, why you need it. In *SIGUCCS '03, ACM*, pages 249–254, 2003.
3. R. Hills. *Common VPN Security Flaws*, 2005. http://www.nta-monitor.com/.
4. T-J Liu, C-Y Chung, and C-L Lee. A high performance and low cost distributed file system. In *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on*, pages 47–50, 2011.
5. M. Melina. *HFS Http File Server*, 2002. http://www.rejetto.com/hfs/.
6. S. Miltchev, J. Smith, V. Prevelakis, A. Keromytis, and S. Ioannidis. Decentralized access control in distributed file systems. *ACM Comput. Surv.*, 40(3):10:1–10:30, August 2008.
7. J. Siegel and P. Lu. User-level remote data access in overlay metacomputers. In *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'02)*, pages 1–4, 2002.
8. J. B. Weissman, M. Marina, and M. Gingras. Optimizing remote file access for parallel and distributed network applications.