RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

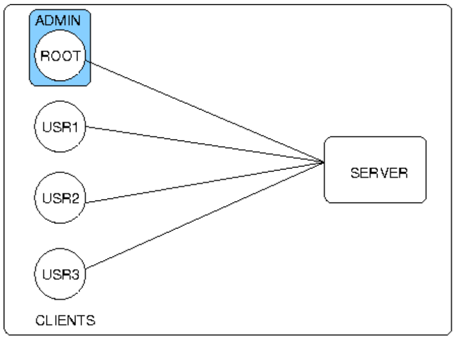
OCLTT
Office of Corporate Liaison and Technology Transfer

Novel Architecture for Controlling File System Access

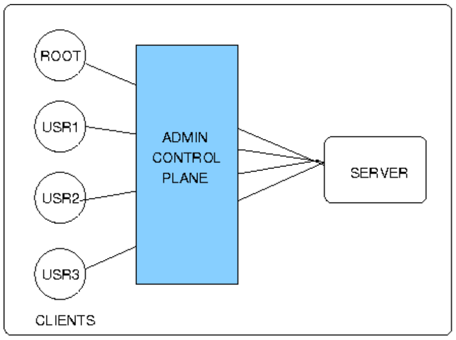
File system administration directly or indirectly affects all users of computer systems today. In an organization, users interact directly with managed file systems, which provide a means of data storage. Users of on-line services, for example, search engines, social networks, photo and video sharing, web based email, etc., rely on managed file systems indirectly, for storing and accessing data. The scale of the user community as well as the complexity of file system access and security policies makes file system administration difficult.

In today’s file system model, administrators have no place. What we call a system administrator is in fact a user with super user (root) privileges. As shown in the first figure (below left), existing approaches limit the effectiveness of administration by simply mapping administrators to root. The root account is powerful. With it, an administrator can access and modify all files, override and modify access permissions, create files anywhere, or delete any file in the file system. Unfortunately, such power turns out to be paradoxically weak. Implementing policies requires sophistication rather than power. Today, even enforcing a simple policy such as time-based access control, e.g. “these files can be accessed only between 9AM and 5PM”, is not trivial to implement correctly. In order to perform the expected duties, file system administrators must become a distinct part of the file system model.

We have designed a new file system architecture that incorporates a separate path (control plane) to the file system, to provide administrators the ability to control the way file systems are accessed. This path, called the Administrator Control Plane (ACP), is distinct from the interface used for accessing files stored on the file system by regular users (data plane). Through ACP, all user activity is visible to the administrators and they are able to automate common tasks (e.g., monitoring, anomaly detection, debugging, etc.) and introduce new arbitrary functionality to extend file systems. The ACP architecture is illustrated in the second figure (below right).



File System with ADMIN as ROOT



File System with ACP

Benefits

- Enhanced Functionality: allows administrators to define extended file system functionality, for users or for administration, without time-consuming modifications to file system or operating system source code.
- Deployable: requires no modifications of existing file systems software. For the case of network file systems, no software agents are required at clients or servers, and deployment is accomplished through the use of a network proxy.
- Centralized Management: represents a single point of management for an administrator’s file system infrastructure, including file system extensions and policies. The greatest impact of this benefit is for large heterogeneous installations of network file systems. Concentrating policy/extension management in a central location reduces the overhead of administration and facilitates rapid file system modification and extension.

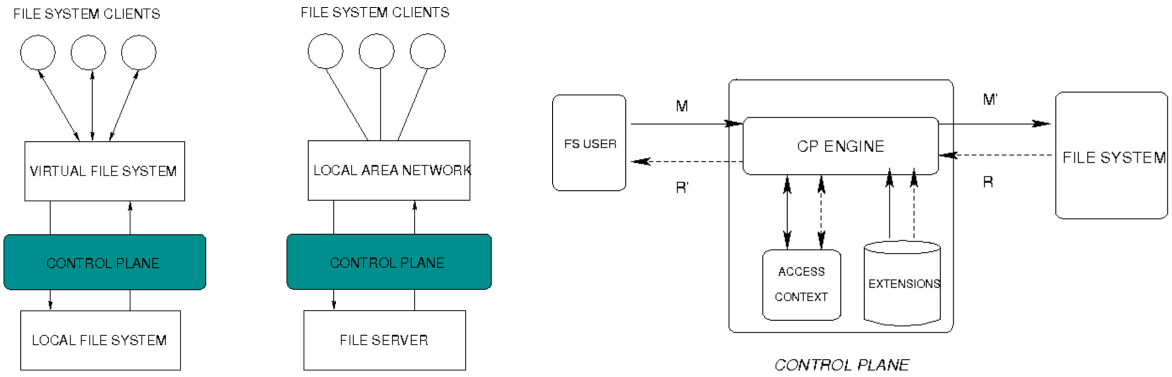
Potential Commercial Use

- Monitoring: collecting various statistics about the health of a file system, its performance, and the ways it is being used. This encompasses a large class of monitoring applications, which includes: statistics gathering, performance monitoring, access logging, user access pattern determination and matching, anomaly detection, etc.
- Access Control: defining alternative file access control policies. These include policies based on dynamic contextual information, such as number of accesses, time, location, user behavior, etc., discretionary policies (e.g., Role-based Access Control (RBAC)), and Mandatory Access Control (MAC) policies.

- Maintenance: eliminating downtime due to administrative maintenance. Through ACP, administrators can implement on-line file system replication (e.g., primary-backup replication), to provide high-availability even during extended periods of file system maintenance.
- Advanced Consistency Models: supporting novel consistency models. Most file systems implement a static unchanging consistency model, which is not configurable. Through ACP, administrators can implement alternate consistency models, for example to ensure simultaneous and atomic updates to groups of files without modifying the file system.
- Semantic Extension: implementing new file system semantics. For example, virtual namespaces for file systems, allowing users on-line mechanisms to sort through the many files they own to quickly find desired files.
- Rapid Extension Prototyping: providing a platform for rapid file system extension prototyping and debugging. Extension developers can quickly implement, install, and evaluate various competing designs without making any changes to the base file system source code. Additionally, they can even execute these extensions in parallel, reducing the time spent in testing and comparing various design trade-offs.

The Technology

The Administrator Control Plane intercepts all file system requests and responses by interposing on the file system data plane. To interpose on the data plane of local file systems all file system operations (requests and responses) are intercepted within the OS kernel as shown in the figure below (left). For network file systems, we introduce a proxy, called *FileWall*, on the network path between file system clients and servers, which intercepts all file system operations as they are sent between the clients and servers as shown in the figure below (center). FileWall interposes on file system accesses to control file system usage and is similar to a firewall, which interposes on network messages to control network usage.



Administrators define file system extensions within ACP using a high-level file system abstraction language, called the control plane language (CPL). By utilizing CPL, and the associated CPL debugger, administrators can concentrate on *what* extensions to implement rather than on *how* to implement them. These extensions are used by administrators to modify and extend file systems, and may be primitive, where the functionality is self-contained, or composite, connecting multiple primitive extensions in a chain. Extensions are executed using the attributes contained in file system operations (requests and responses), in the context of state maintained by the extension or present in the execution environment, called the *access contex*t. Additionally, extensions may choose to store access context in persistent storage through a database which guarantees ACID properties (Atomicity, Consistency, Integrity, and Durability). Finally, reuse of code is supported through libraries of extensions.

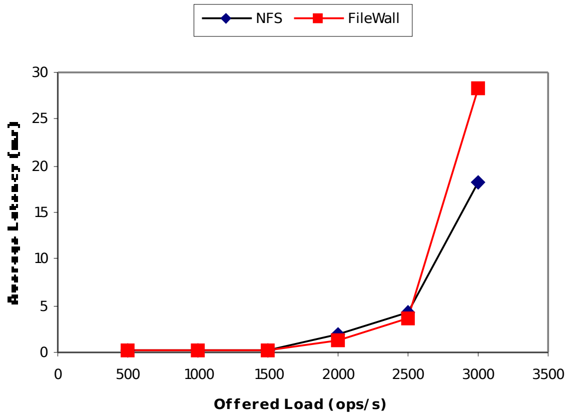
The primary control function provided by ACP is operation transformation. To illustrate operation transformation, the figure above (right) shows the flow of a user request message (M) and the response message (R) through the ACP. The request is issued, by a user thread of execution, through the standard file system interface. The ACP intercepts the operation and invokes extensions that use access context to transform this operation to M' and passes M' to the file system. The file system returns a response (R), which is transformed by the ACP to R' and passed to the user thread as the file system response. There are three types of operation transformations that can be specified: attribute transformation, flow transformation, and flow coordination. These primitives represent the minimal set of functionality supported by the control plane for operation transformation. Since the control plane is isolated from the file system data plane, all file system extensions are implemented in the control plane using a combination of these three primitives:

- Attribute Transformation: File system operations are constructed as a set of arguments in file system requests and responses. Extensions may modify these attributes to affect the action performed at either end (user or file system). Since data updates (writes) and data queries (reads) are included as arguments to file system operations, data transformations can also be performed with this primitive.

- Flow Transformation: A flow in a file system represents a channel where all file system operations are exchanged between user threads and the file system. Flow transformation modifies file system behavior using operation filtering, operation reordering, and operation injection. Filtering removes operations sent by one end of a flow from being received by the other end. Injection introduces additional operations into a flow. Finally, reordering modifies the sequence of operations within a flow.
- Flow Coordination: When ACP extensions affect more than one flow, additional mechanisms to transform operations across a collection of flows is necessary. Flow coordination includes multiplexing and demultiplexing across a group of flows. Multiplexing combines multiple flows of operations into a single flow and demultiplexing separates a single flow of operations into multiple flows. These multiple flows may correspond to a previously multiplexed collection or may be new ACP generated flows (e.g., for replication of a flow).

Preliminary Prototype and Results

We have implemented a preliminary prototype FileWall for network file systems, and have studied the feasibility and performance of this prototype system. FileWall is implemented in the Click Modular Router framework (http://www.read.cs.ucla.edu/click) as an external user-level package, and the access context is implemented using the open source Berkeley DB (http://dev.sleepycat.com). A technical report that presents the full set of results from our prototype evaluation is available. The following summarizes our findings.



In our experimental setup, all systems are Dell Poweredge 2600 SMP systems with two 2.4GHz Intel Pentium II Xeon CPUs, 2GB of RAM, and 36GB 15K RPM SCSI drives. The figure above shows the performance results comparing FileWall against the default NFS, while executing the FStress benchmark. We observe that FileWall latencies are comparable to default NFS up to an offered load of 2,500 requests/s. The systems diverge beyond this point. The increase in observed latency with FileWall is small (around 15%) and the enhanced functionality provided in exchange by FileWall make them acceptable. For both cases, the NFS server is overloaded at around 3,000 requests/s.

Research and Licensing

All of the intellectual property associated with the Administrator Control Plane technology (including FileWall) is available for licensing. Additional research and development support is available from Prof. Liviu Iftode's research group (DiscoLab) in the Computer Science Department at Rutgers University. A complete technical report is available at the DiscoLab website – http://www.cs.rutgers.edu/discolab.

Patent Information

” FileWall: An Architecture for Controlling Access to Network File Systems”, **Rutgers Docket 07-045**, Patent Pending

Contact

Andrew Staroscik, Manager Licensing and Technology
Office of Corporate Liaison and Technology Transfer, Rutgers University - http://ocltt.rutgers.edu
ASB III, 3 Rutgers Plaza, New Brunswick, NJ 08901
Phone: 732-932-0115 x3010 Fax: 732-932-0146 E-mail: staroscik@ocltt.rutgers.edu