Abstract

Although several UNIX techniques for sharing memory exist, such as System V IPC Shared Memory and POSIX Shared Memory, these schemes are not suitable for use internal to the operating system and cannot protect against all threats posed by memory sharing. Therefore, we propose a novel, flexible memory-protection scheme for UNIX-like systems based on fine-grained, delegatable memory grants. We have already applied our ideas to safe memory copies and are currently investigating extensions for direct memory access (DMA) and memory mapping. Although memory grants originated in the MINIX 3 operating system, they are generally applicable and can be easily ported to other platforms.

1 Introduction

This work presents a novel, flexible memory-protection scheme for UNIX-like systems based on memory grants. A memory grant is a mechanism for sharing memory, similar to a capability in that it can be transferred to another party in order to grant fine-grained access. Once a memory grant has been received, the access rights may be delegated to third parties at the grantee’s discretion. The proposed structure also supports immediate revocation of memory grants at the grantor’s discretion.

We propose a generalized memory-protection model based on memory grants. Grants originate from the MINIX 3 operating system where they were used to enable safe memory copies from and to device drivers [2]. On top of this, we propose two new extensions that bring memory grant-based protection for memory mappings and direct memory access (DMA).

Memory grants are not tied to device drivers or MINIX 3, but, in fact, represent a generally applicable model. For example, memory grants also could be used to achieve fine-grained protection for ordinary applications. In addition, memory grants could be ported to other operating systems including Linux and Windows.

2 Contribution of Grants

While several memory-sharing schemes exist, none of the existing models provide the same flexibility and degree of protection as memory grants do. Current UNIX techniques such as System V IPC and POSIX Shared Memory have several problems that are addressed by memory grants:

- Protection is based on group ID and/or user ID and cannot grant access to individual processes.
- Course granularity based on page size, even if only small data structures are to be shared.
- Delegation of rights to access a given piece of memory is not supported.
- Access rights are not automatically invalidated if a process sharing memory crashes.
- Not suitable for low-level drivers that cannot interface with the high-level POSIX servers.
- Cannot be used for safe direct memory access (DMA) because I/O-MMU integration is lacking.

In addition, a number of more specialized approaches exist, including seL4’s frame capabilities [1]. Although several parallels can be drawn in terms of functionality offered, the naming scheme, usage, and implementation techniques are different from memory grants.

3 Structure of Memory Grants

The structure of a memory grant is given in Fig. 1. The flags field indicates whether the grant is in use, the grant’s type, and the kind of access allowed. A direct grant means that a process A grants another process B limited access to a memory area in its own address space. The receiver of a direct grant, say, B, is allowed to transfer its access rights to a third process C by means of an indirect grant. The memory area covered by an indirect grant is always relative to the previous grant, which can either be a direct grant or indirect grant. Finally, the R/W flags define the access type that is granted: read, write or both.
Although we are still extending the memory grant model and have not yet finalized the API, we envision memory-grant operations in the following categories:

- Grant table management
- Grant creation and revocation
- Grant permission lookup
- Memory copying
- Memory mapping
- Direct memory access

These operations will be implemented in a system library. The library code performs sanity checks on the grant table, grant ID and memory grant and then determines which operation is requested.

5 Performance Considerations

Memory grants do not come with an inherent performance overhead. Most grant table management, such as grant creation and revocation, can be done local to the caller. Privileged operations will have to be mediated by the kernel in order to enforce the protection, but setting up memory mappings and allowing DMA access can be done once during initialization and are not in the critical path. While this is the case for memory copies, the costs are limited to a single context switch, since the kernel can directly access all physical memory to read from the grant tables; no context switching is needed to follow the chain. Moreover, resolving a memory grant will be limited to one or two table lookups in typical usage scenarios.

6 Work in Progress

A prototype of the full memory-grant model is currently a work in progress. Safe memory copies could be taken over from our previous work [2], but memory mapping and direct memory are still to be done. Since these operations separate the time of checking from the time when the memory access takes place, several problems with respect to grant revocation need to be solved. We have outlined a possible solution and intend to implement it in the MINIX 3 operating system.

REFERENCES

Memory Sharing Revisited

Jorrit N. Herder, Herbert Bos, Arun Thomas, Ben Gras, Andrew S. Tanenbaum
jnherder,herbertb,arun,beng,ast}@cs.vu.nl
Vrije Universiteit, Amsterdam, The Netherlands

Shortcomings of Existing Protection Schemes

System V IPC and POSIX Shared Memory lack flexibility and offer limited memory protection. Some shortcomings include:

- Coarse-grained, page-based protection
- Protection based on UID not process
- Access rights cannot be delegated
- No seamless integration for safe DMA

Fine-grained, Delegatable Memory Grants

Process that wants to share memory creates a grant table, builds a memory grant, and sends index to other party: (proc ID, grant ID) identifies the grant.

Recipient of a memory grant must call the kernel in order to perform privileged grant operations. Kernel validates the access rights and performs the request.

Definition of API and implementation of operations are a work in progress:

- Grant creation and revocation
- Memory copying
- Memory mapping
- Direct memory access (DMA)

Take-home Messages

- Memory grants are a novel alternative to existing memory protection models.
- Precise per-process access control for byte-granularity memory regions.
- Used in MINIX 3 to protect against memory corruption by buggy drivers.

Supported by the Netherlands Organization for Scientific Research (NWO) under grant 612-060-420.