SUSE

Fanotify

Linux file system notification subsystem

Jan Kára <jack@suse.cz>
SUSE

Overview

A bit of history
Basic fanotify functionality
Permission events
File system-wide monitoring
HSM events
Future outlook

A bit of history

Motivation

- Provide efficient notification about events (primarily changes) happening in the file system
  - Useful for desktop search, file system backup, file open dialogues, ...
  - Speeds up blunt approaches like repeatedly calling stat(2)
- Fanotify also allows watching for and mediating file access and some other file system events

History

Dnotify -> inotify -> fanotify

- Dnotify
  - Merged during 2.4 times
  - Only notification about changes in directories
  - Require open directory
  - Notification through signal (something has changed)
- Inotify
  - Merged in 2.6.13 (2005)
  - Monitoring files & directories
  - No need to keep files or directories open
  - Problems with watching large directory hierarchies and identifying changed objects

History

Dnotify -> inotify -> fanotify

- Fanotify
  - Merged in 2.6.36 (2010)
  - Originally driven by needs of antivirus scanners
  - Trying to address problems with watching large hierarchies and accessing file / directory referenced in the event

Basic fanotify functionality

Initialization

- fanotify_init(flags, event_f_flags)
  - Creates notification group and returns a file descriptor for it
  - Full functionality requires CAP_SYS_ADMIN, inotify-like features available without it
  - *flags* specify:
    - Type of notification group (FAN_CLASS_NOTIF, FAN_CLASS_CONTENT, FAN_CLASS_PRE_CONTENT)
    - Information returned with each notification event (FAN_REPORT_FID, FAN_REPORT_NAME, ...)
    - Notification group limits (FAN_UNLIMITED_QUEUE, FAN_UNLIMITED_MARKS)

Placing notification marks

- fanotify_mark(fanotify_fd, flags, mask, dirfd, pathname)
  - Places notification mark on file, directory, mountpoint
  - Originally supported events: FAN_ACCESS, FAN_MODIFY, FAN_OPEN, FAN_OPEN_EXEC, FAN_CLOSE_NOWRITE, FAN_CLOSE_WRITE
  - Not useful much for directories
  - FAN_EVENT_ON_CHILD in mask – report events for immediate children of a directory
  - Ignore masks (FAN_MARK_IGNORE_MASK flag)
    - Allows to ignore specified events on file / directory / mount
    - By default gets cleared on first modification event – need FAN_MARK_IGNORED_SURV_MODIFY

Receiving events

- Read from notification group file descriptor yields:

```
struct fanotify_event_metadata {
  __u32 event_len;
  __u8 vers;
  __u8 reserved;
  __u16 metadata_len;
  __u64 mask;
  __s32 fd;
  __s32 pid;
}
```

Permission events

- FAN_OPEN_PERM, FAN_ACCESS_PERM, FAN_OPEN_EXEC_PERM (since 5.0)
- Allow mediating access to files (antivirus scanners use this)
  - Syscall is paused until we get reply to fanotify event. Based on reply syscall continues or returns with error
- Reply is sent to kernel by writing to notification group descriptor:

```
struct fanotify_response {
  __s32 fd;
  __u32 response;
}
```

Better filesystem monitoring

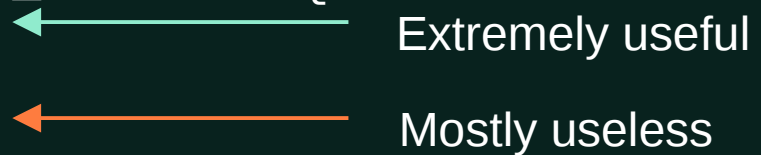
Filesystem wide monitoring

- New mark type: FAN_MARK_FILESYSTEM (since 4.20)
  - Watching full filesystem independently of a mountpoint
  - Cannot use file descriptor to identify file from which event originates
  - File identified with a fsid + file handle pair
    - All filesystems extended to provide a meaningful fsid and ability to get a file handle for an inode

Supplemental event information

- Additional information reported with events

```
struct fanotify_event_metadata {
  __u32 event_len;
  __u8 vers;
  __u8 reserved;
  __u16 metadata_len;
  __u64 mask;
  __s32 fd;
  __s32 pid;
}
```



Supplemental event information

- Needs to be explicitly enabled on fanotify_init(2) time
  - FAN_REPORT_FID, FAN_REPORT_DIR_FID, FAN_REPORT_TARGET_FID, FAN_REPORT_NAME
- Supplemental information appended after initial event structure
  - event_len in initial structure shows total length
  - Each supplemental record has a header:

```
struct fanotify_event_info_header {
  __u8 info_type;
  __u8 pad;
  __u16 len;
}
```

FAN_REPORT_FID

- Return fsid + file handle to identify file / directory in event
- Used instead of open file descriptor
  - No need to open files – faster, does not pin files in memory
  - Works for unprivileged users
  - Works for all types of events
  - Users with CAP_DAC_READ_SEARCH can use open_by_handle(2) to open

```
struct fanotify_event_info_fid {
  struct fanotify_event_info_header hdr;
  __kernel_fsid_t fsid;
  unsigned char handle[0];
}
```

FAN_REPORT_DIR_FID

- Similar to FAN_REPORT_FID but reports directory associated with the event
  - Reports parent directory for events on files
  - Identical to FAN_REPORT_FID for events on directories
  - Sometimes not available (e.g. for unlinked inodes or root directory)
- Useful for reconstructing paths where event happened

FAN_REPORT_NAME

- Reports name of object inside a directory identified by DIR_FID
  - Requires FAN_REPORT_DIR_FID
  - Name is “.” for events on directories
  - open_by_handle_at(2) can be used to open the object

Directory events (since 5.1)

- Events informing about changes to directories
- FAN_CREATE, FAN_DELETE, FAN_DELETE_SELF, FAN_ATTRIB, FAN_MOVED_FROM, FAN_MOVED_TO, FAN_MOVED_SELF, FAN_RENAME
  - FAN_RENAME combines information about source and target directories and names
  - FAN_REPORT_TARGET_FID may be used to provide information about child of the directory affected by the event
- Allows efficient monitoring of large directory hierarchy for changes
- With these events fanotify achieved feature parity with inotify

Evictable marks (since 5.19)

- Normally, notification marks pin inodes in memory
  - Impractical when we need to mark lots of inodes (e.g. with ignore marks)
- Mark can be created with FAN_MARK_EVICTABLE flag to avoid pinning inode
  - Mark is freed when inode is freed from memory

Hierarchical storage management using fanotify

Photo by: Gerhard Marcks

HSM Basics

- Fill in filesystem on fast storage on demand from slow storage
  - Local cache of website tree
  - Local cache of a network filesystem
- Slow storage access transparent
- Approaches
  - Fill file contents on open
  - Fill file contents on read / write…
  - Fill directory contents on open
- Existing Linux solutions
  - FUSE – very flexible, hard to achieve “local filesystem” performance for cached data
  - Fscache – tied to couple network filesystems

Fanotify pre-content events (staged for 6.12)

- Kind of similar to permission events
  - System call is blocked until userspace replies to the event
- Require the highest FAN_CLASS_PRE_CONTENT notification class of the notification group
  - Which means CAP_SYS_ADMIN is required
- FAN_PRE_ACCESS and FAN_PRE_MODIFY events
  - Generated on read, write (including countless variants), open but also during page faults
  - Events include IO range
  - Generated outside of locks (including fs freezing prevention)
- Currently enabled for xfs, btrfs, ext4, bcachefs
  - Enabling for other filesystems is mostly a matter of demand and testing

HSM implementation using fanotify

- Daemon watching for FAN_PRE_ACCESS / FAN_PRE_MODIFY events on the whole filesystem
  - Keeps track of filled in content for a file
  - Fills in content for given range when receiving event before acknowledging it
  - Needs to be careful to fill in content without generating events
    - Separate mount for filling in with ignore mark
  - Once the whole file is filled in, inode can be marked with ignore mark
- Data consistency relies on the daemon running
  - Plan to block filesystem access unless HSM events are handled
- Implementation (with preliminary patches) already used in a few deployments

Outlook

Outlook

What you might want to work on ;-)

- Loose ends with hierarchical storage management
  - Making filesystem safe in case of daemon crashes
  - Handling of directories
- More efficient watching of directory subtrees
  - eBPF ignore marks?
  - Marks which automatically copy themselves from the parent?
- Persistent change notification service

SUSE

Thank you

For more information, contact SUSE at:

+1 800 796 3700 (U.S./Canada)

Frankenstrasse 146

90461 Nürnberg

www.suse.com