

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-08/msg02737.html>

- *From:* Mathieu Desnoyers <compudj@xxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 6 Aug 2008 13:32:25 -0400
-

Hi,

I looks like this patch (commit 20d8b67c06fa5e74f44e80b0a0fd68c8327f7c6a) broke LTTng. LTTng expects its buffer file creation callback to be called when it calls relay_open. Actually, it passes a filename to the relay_open, so I don't see why the create_buf_file callback is not being called anymore. I'll revert it in my LTTng tree for now.

I'll eventually need a new create_buf() callback to correctly handle this and allow LTTng to do early boot tracing. When this new callback gets called, I can setup the data structures I use to manage my internal buffer offsets. Then the already existing create_buf_file callback could just do the debugfs calls.

Mathieu

* Eduard – Gabriel Munteanu (eduard.munteanu@xxxxxxxxxxxx) wrote:

Allows one to create and use a channel with no associated files. Files can be initialized later. This is useful in scenarios such as logging in early code, before VFS is up. Therefore, such channels can be created and used as soon as kmem_cache_init() completed.

This is needed by kmemtrace to do tracing in early kernel code.

Signed-off-by: Eduard – Gabriel Munteanu <eduard.munteanu@xxxxxxxxxxxx>

Documentation/filesystems/relay.txt | 11 +++
include/linux/relay.h | 5 +
kernel/relay.c | 170 ++++++-----
3 files changed, 157 insertions(+), 29 deletions(-)

diff --git a/Documentation/filesystems/relay.txt b/Documentation/filesystems/relay.txt
index 094f2d2..b417f83 100644
--- a/Documentation/filesystems/relay.txt
+++ b/Documentation/filesystems/relay.txt
@@ -161,6 +161,7 @@ TBD(curr. line MT:/API)

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

```
relay_close(chan)
relay_flush(chan)
relay_reset(chan)
+ relay_late_setup_files(chan, base_filename, parent)
```

channel management typically called on instigation of userspace:

@@ -294,6 +295,16 @@ user-defined data with a channel, and is immediately available (including in create_buf_file()) via chan->private_data or buf->chan->private_data.

+Buffer-only channels

+-----

+

+These channels have no files associated and can be created with +relay_open(NULL, NULL, ...). Such channels are useful in scenarios such +as when doing early tracing in the kernel, before the VFS is up. In these +cases, one may open a buffer-only channel and then call +relay_late_setup_files() when the kernel is ready to handle files, +to expose the buffered data to the userspace.

+

Channel 'modes'

diff --git a/include/linux/relay.h b/include/linux/relay.h

index 6cd8c44..953fc05 100644

--- a/include/linux/relay.h

+++ b/include/linux/relay.h

@@ -48,6 +48,7 @@ struct rchan_buf

size_t *padding; /* padding counts per sub-buffer */

size_t prev_padding; /* temporary variable */

size_t bytes_consumed; /* bytes consumed in cur read subbuf */

+ size_t early_bytes; /* bytes consumed before VFS initd */

unsigned int cpu; /* this buf's cpu */

} ____cacheline_aligned;

@@ -68,6 +69,7 @@ struct rchan

int is_global; /* One global buffer ? */

struct list_head list; /* for channel list */

struct dentry *parent; /* parent dentry passed to open */

+ int has_base_filename; /* has a filename associated? */

char base_filename[NAME_MAX]; /* saved base filename */

};

@@ -169,6 +171,9 @@ struct rchan *relay_open(const char *base_filename,

size_t n_subbufs,

struct rchan_callbacks *cb,

void *private_data);

+extern int relay_late_setup_files(struct rchan *chan,

+ const char *base_filename,

+ struct dentry *parent);

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

```
extern void relay_close(struct rchan *chan);
extern void relay_flush(struct rchan *chan);
extern void relay_subbufs_consumed(struct rchan *chan,
diff --git a/kernel/relay.c b/kernel/relay.c
index 7de644c..4387b35 100644
--- a/kernel/relay.c
+++ b/kernel/relay.c
@@ -407,6 +407,35 @@ void relay_reset(struct rchan *chan)
}
EXPORT_SYMBOL_GPL(relay_reset);

+static inline void relay_set_buf_dentry(struct rchan_buf *buf,
+ struct dentry *dentry)
+{
+ buf->dentry = dentry;
+ buf->dentry->d_inode->i_size = buf->early_bytes;
+}
+
+static struct dentry *relay_create_buf_file(struct rchan *chan,
+ struct rchan_buf *buf,
+ unsigned int cpu)
+{
+ struct dentry *dentry;
+ char *tmpname;
+
+ tmpname = kzalloc(NAME_MAX + 1, GFP_KERNEL);
+ if (!tmpname)
+ return NULL;
+ snprintf(tmpname, NAME_MAX, "%s%d", chan->base_filename, cpu);
+
+ /* Create file in fs */
+ dentry = chan->cb->create_buf_file(tmpname, chan->parent,
+ S_IRUSR, buf,
+ &chan->is_global);
+
+ kfree(tmpname);
+
+ return dentry;
+}
+
+/*
+ * relay_open_buf - create a new relay channel buffer
+ */
@@ -416,45 +445,34 @@ static struct rchan_buf *relay_open_buf(struct rchan *chan,
unsigned int cpu)
{
struct rchan_buf *buf = NULL;
struct dentry *dentry;
- char *tmpname;

if (chan->is_global)
```

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

```
return chan->buf[0];

- tmpname = kzalloc(NAME_MAX + 1, GFP_KERNEL);
- if (!tmpname)
- goto end;
- snprintf(tmpname, NAME_MAX, "%s%d", chan->base_filename, cpu);
-
buf = relay_create_buf(chan);
if (!buf)
- goto free_name;
+ return NULL;
+
+ if (chan->has_base_filename) {
+ dentry = relay_create_buf_file(chan, buf, cpu);
+ if (!dentry)
+ goto free_buf;
+ relay_set_buf_dentry(buf, dentry);
+ }

buf->cpu = cpu;
__relay_reset(buf, 1);

- /* Create file in fs */
- dentry = chan->cb->create_buf_file(tmpname, chan->parent, S_IRUSR,
- buf, &chan->is_global);
- if (!dentry)
- goto free_buf;
-
- buf->dentry = dentry;
-
if(chan->is_global) {
chan->buf[0] = buf;
buf->cpu = 0;
}

- goto free_name;
+ return buf;

free_buf:
relay_destroy_buf(buf);
- buf = NULL;
-free_name:
- kfree(tmpname);
-end:
- return buf;
+ return NULL;
}

/**
@@ -537,8 +555,8 @@ static int __cpuinit relay_hotcpu_callback(struct notifier_block *nb,
```

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

```
/**
 * relay_open – create a new relay channel
 * – * @base_filename: base name of files to create
 * – * @parent: dentry of parent directory, %NULL for root directory
 * + * @base_filename: base name of files to create, %NULL for buffering only
 * + * @parent: dentry of parent directory, %NULL for root directory or buffer
 * * @subbuf_size: size of sub-buffers
 * * @n_subbufs: number of sub-buffers
 * * @cb: client callback functions
 * @@ –560,8 +578,6 @@ struct rchan *relay_open(const char *base_filename,
 * {
 * unsigned int i;
 * struct rchan *chan;
 * – if (!base_filename)
 * – return NULL;
 *
 * if (!(subbuf_size && n_subbufs))
 * return NULL;
 * @@ –576,7 +592,10 @@ struct rchan *relay_open(const char *base_filename,
 * chan->alloc_size = FIX_SIZE(subbuf_size * n_subbufs);
 * chan->parent = parent;
 * chan->private_data = private_data;
 * – strcpy(chan->base_filename, base_filename, NAME_MAX);
 * + if (base_filename) {
 * + chan->has_base_filename = 1;
 * + strcpy(chan->base_filename, base_filename, NAME_MAX);
 * + }
 * setup_callbacks(chan, cb);
 * kref_init(&chan->kref);
 *
 * @@ –604,6 +623,94 @@ free_bufs:
 * }
 * EXPORT_SYMBOL_GPL(relay_open);
 *
 * +struct rchan_percpu_buf_dispatcher {
 * + struct rchan_buf *buf;
 * + struct dentry *dentry;
 * +};
 * +
 * +/* Called in atomic context. */
 * +static void __relay_set_buf_dentry(void *info)
 * +{
 * + struct rchan_percpu_buf_dispatcher *p = info;
 * +
 * + relay_set_buf_dentry(p->buf, p->dentry);
 * +}
 * +
 * +/**
 * + * relay_late_setup_files – triggers file creation
 * + * @chan: channel to operate on
 * + * @base_filename: base name of files to create
```

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

```
+ * @parent: dentry of parent directory, %NULL for root directory
+ *
+ * Returns 0 if successful, non-zero otherwise.
+ *
+ * Use to setup files for a previously buffer-only channel.
+ * Useful to do early tracing in kernel, before VFS is up, for example.
+ */
+int relay_late_setup_files(struct rchan *chan,
+ const char *base_filename,
+ struct dentry *parent)
+{
+ int err = 0;
+ unsigned int i, curr_cpu;
+ unsigned long flags;
+ struct dentry *dentry;
+ struct rchan_percpu_buf_dispatcher disp;
+
+ if (!chan || !base_filename)
+ return -EINVAL;
+
+ strncpy(chan->base_filename, base_filename, NAME_MAX);
+
+ mutex_lock(&relay_channels_mutex);
+ /* Is chan already set up? */
+ if (unlikely(chan->has_base_filename))
+ return -EEXIST;
+ chan->has_base_filename = 1;
+ chan->parent = parent;
+ curr_cpu = get_cpu();
+ /*
+ * The CPU hotplug notifier ran before us and created buffers with
+ * no files associated. So it's safe to call relay_setup_buf_file()
+ * on all currently online CPUs.
+ */
+ for_each_online_cpu(i) {
+ if (unlikely(!chan->buf[i])) {
+ printk(KERN_ERR "relay_late_setup_files: CPU %u "
+ "has no buffer, it must have!\n", i);
+ BUG();
+ err = -EINVAL;
+ break;
+ }
+
+ dentry = relay_create_buf_file(chan, chan->buf[i], i);
+ if (unlikely(!dentry)) {
+ err = -EINVAL;
+ break;
+ }
+
+ if (curr_cpu == i) {
+ local_irq_save(flags);
```

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

```
+ relay_set_buf_dentry(chan->buf[i], dentry);
+ local_irq_restore(flags);
+ } else {
+ disp.buf = chan->buf[i];
+ disp.dentry = dentry;
+ smp_mb();
+ /* relay_channels_mutex must be held, so wait. */
+ err = smp_call_function_single(i,
+ __relay_set_buf_dentry,
+ &disp, 0, 1);
+ }
+ if (unlikely(err))
+ break;
+ }
+ put_cpu();
+ mutex_unlock(&relay_channels_mutex);
+
+ return err;
+}
+
+/**
+ * relay_switch_subbuf - switch to a new sub-buffer
+ * @buf: channel buffer
+ @@ -627,8 +734,13 @@ size_t relay_switch_subbuf(struct rchan_buf *buf, size_t length)
+ old_subbuf = buf->subbufs_produced % buf->chan->n_subbufs;
+ buf->padding[old_subbuf] = buf->prev_padding;
+ buf->subbufs_produced++;
+ - buf->dentry->d_inode->i_size += buf->chan->subbuf_size -
+ - buf->padding[old_subbuf];
+ + if (buf->dentry)
+ + buf->dentry->d_inode->i_size +=
+ + buf->chan->subbuf_size -
+ + buf->padding[old_subbuf];
+ + else
+ + buf->early_bytes += buf->chan->subbuf_size -
+ + buf->padding[old_subbuf];
+ smp_mb();
+ if (waitqueue_active(&buf->read_wait))
+ /*
+ @@ -1237,4 +1349,4 @@ static __init int relay_init(void)
+ return 0;
+ }
+
+ -module_init(relay_init);
+ +early_initcall(relay_init);
+ --
+ 1.5.5.4
```

Re: [PATCH 3/3] relay: Add buffer-only channels; useful for early logging.

Mathieu Desnoyers

OpenPGP key fingerprint: 8CD5 52C3 8E3C 4140 715F BA06 3F25 A8FE 3BAE 9A68

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>