

[PATCH 4/5] jffs2: Add a "favourlzo" compression mode to jffs2

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg10182.html>

- From: Richard Purdie <rpurdie@xxxxxxxxxxxxxxxxx>
- Date: Wed, 28 Feb 2007 19:13:48 +0000

Add a "favourlzo" compression mode to jffs2 which tries to optimise by size but gives lzo an advantage when comparing sizes. This means the faster lzo algorithm can be preferred when there isn't much difference in compressed size (the exact threshold can be changed).

Signed-off-by: Richard Purdie <rpurdie@xxxxxxxxxxxxxxxxx>

```

---
fs/Kconfig | 7 +++++++
fs/jffs2/compr.c | 51 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-----
fs/jffs2/compr.h | 3 +++
3 files changed, 56 insertions(+), 5 deletions(-)

```

Index: linux/fs/Kconfig

```
=====
```

```

--- linux.orig/fs/Kconfig 2007-02-28 18:12:31.000000000 +0000
+++ linux/fs/Kconfig 2007-02-28 18:12:33.000000000 +0000
@@ -1359,6 +1359,13 @@ config JFFS2_CMODE_SIZE
Tries all compressors and chooses the one which has the smallest
result.

```

```

+config JFFS2_CMODE_FAVOURLZO
+ bool "Favour LZO"
+ help
+ Tries all compressors and chooses the one which has the smallest
+ result but gives some preference to LZO (which has faster
+ decompression) at the expense of size.
+
+ endchoice

```

config CRAMFS

Index: linux/fs/jffs2/compr.c

```
=====
```

```

--- linux.orig/fs/jffs2/compr.c 2007-02-28 18:12:31.000000000 +0000
+++ linux/fs/jffs2/compr.c 2007-02-28 18:13:09.000000000 +0000
@@ -26,6 +26,34 @@ static int jffs2_compression_mode = JFFS
/* Statistics for blocks stored without compression */

```

[PATCH 4/5] jffs2: Add a "favourlzo" compression mode to jffs2

```
static uint32_t none_stat_compr_blocks=0,none_stat_decompr_blocks=0,none_stat_compr_size=0;

+
+/*
+ * Return 1 to use this compression
+ */
+static int jffs2_is_best_compression(struct jffs2_compressor *this,
+ struct jffs2_compressor *best, uint32_t size, uint32_t bestsize)
+{
+ switch (jffs2_compression_mode) {
+ case JFFS2_COMPR_MODE_SIZE:
+ if (bestsize > size)
+ return 1;
+ return 0;
+ case JFFS2_COMPR_MODE_FAVOURLZO:
+ if ((this->compr == JFFS2_COMPR_LZO) && (bestsize > size))
+ return 1;
+ if ((best->compr != JFFS2_COMPR_LZO) && (bestsize > size))
+ return 1;
+ if ((this->compr == JFFS2_COMPR_LZO) && (bestsize > (size * FAVOUR_LZO_PERCENT / 100)))
+ return 1;
+ if ((bestsize * FAVOUR_LZO_PERCENT / 100) > size)
+ return 1;
+
+ return 0;
+ }
+ /* Shouldn't happen */
+ return 0;
+}
+
+/* jffs2_compress:
+ * @data: Pointer to uncompressed data
+ * @cdata: Pointer to returned pointer to buffer for compressed data
+ @@ -91,6 +119,7 @@ uint16_t jffs2_compress(struct jffs2_sb_
+ if (ret == JFFS2_COMPR_NONE) kfree(output_buf);
+ break;
+ case JFFS2_COMPR_MODE_SIZE:
+ case JFFS2_COMPR_MODE_FAVOURLZO:
+ orig_slen = *datalen;
+ orig_dlen = *cdatalen;
+ spin_lock(&jffs2_compressor_list_lock);
+ @@ -99,7 +128,7 @@ uint16_t jffs2_compress(struct jffs2_sb_
+ if ((!this->compress)||(!this->disabled))
+ continue;
+ /* Allocating memory for output buffer if necessary */
+ - if ((this->compr_buf_size<orig_dlen)&&(this->compr_buf)) {
+ + if ((this->compr_buf_size<orig_slen)&&(this->compr_buf)) {
+ spin_unlock(&jffs2_compressor_list_lock);
+ kfree(this->compr_buf);
+ spin_lock(&jffs2_compressor_list_lock);
+ @@ -108,15 +137,15 @@ uint16_t jffs2_compress(struct jffs2_sb_
```

[PATCH 4/5] jffs2: Add a "favourlzo" compression mode to jffs2

```

}
if (!this->compr_buf) {
spin_unlock(&jffs2_compressor_list_lock);
- tmp_buf = kmalloc(orig_dlen,GFP_KERNEL);
+ tmp_buf = kmalloc(orig_slen,GFP_KERNEL);
spin_lock(&jffs2_compressor_list_lock);
if (!tmp_buf) {
- printk(KERN_WARNING "JFFS2: No memory for compressor allocation. (%d bytes)\n",orig_dlen);
+ printk(KERN_WARNING "JFFS2: No memory for compressor allocation. (%d bytes)\n",orig_slen);
continue;
}
else {
this->compr_buf = tmp_buf;
- this->compr_buf_size = orig_dlen;
+ this->compr_buf_size = orig_slen;
}
}
this->usecount++;
@@ -127,7 +156,8 @@ uint16_t jffs2_compress(struct jffs2_sb_
spin_lock(&jffs2_compressor_list_lock);
this->usecount--;
if (!compr_ret) {
- if ((!best_dlen)||((best_dlen>*cdatalen)) {
+ if (((!best_dlen) || jffs2_is_best_compression(this, best, *cdatalen, best_dlen))
+ && (*cdatalen < *datalen)) {
best_dlen = *cdatalen;
best_slen = *datalen;
best = this;
@@ -326,6 +356,8 @@ char *jffs2_get_compression_mode_name(vo
return "priority";
case JFFS2_COMPR_MODE_SIZE:
return "size";
+ case JFFS2_COMPR_MODE_FAVOURLZO:
+ return "favourlzo";
}
return "unkown";
}
@@ -344,6 +376,10 @@ int jffs2_set_compression_mode_name(cons
jffs2_compression_mode = JFFS2_COMPR_MODE_SIZE;
return 0;
}
+ if (!strcmp("favourlzo", name, 9)) {
+ jffs2_compression_mode = JFFS2_COMPR_MODE_FAVOURLZO;
+ return 0;
+ }
return 1;
}

@@ -437,9 +473,14 @@ int __init jffs2_compressors_init(void)
jffs2_compression_mode = JFFS2_COMPR_MODE_SIZE;
D1(printk(KERN_INFO "JFFS2: default compression mode: size\n");)

```

[PATCH 4/5] jffs2: Add a "favourlzo" compression mode to jffs2

```
#else
+ #ifdef CONFIG_JFFS2_CMODE_FAVOURLZO
+ jffs2_compression_mode = JFFS2_COMPR_MODE_FAVOURLZO;
+ D1(printk(KERN_INFO "JFFS2: default compression mode: favourlzo\n"));
+ #else
D1(printk(KERN_INFO "JFFS2: default compression mode: priority\n"));
#endif
#endif
+ #endif
return 0;
}
```

Index: linux/fs/jffs2/compr.h

```
=====
--- linux.orig/fs/jffs2/compr.h 2007-02-28 18:12:31.000000000 +0000
+++ linux/fs/jffs2/compr.h 2007-02-28 18:12:33.000000000 +0000
@@ -41,6 +41,9 @@
#define JFFS2_COMPR_MODE_NONE 0
#define JFFS2_COMPR_MODE_PRIORITY 1
#define JFFS2_COMPR_MODE_SIZE 2
+ #define JFFS2_COMPR_MODE_FAVOURLZO 3
+
+ #define FAVOUR_LZO_PERCENT 80
```

```
struct jffs2_compressor {
struct list_head list;
```

—

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

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

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