

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-12/msg08019.html>

- From: Dan Williams <dan.j.williams@xxxxxxxxxx>
- Date: Fri, 21 Dec 2007 18:06:48 -0700

The tx_set_src and tx_set_dest methods were originally implemented to allow an array of addresses to be passed down from async_xor to the dmaengine driver while minimizing stack overhead. Removing these methods allows drivers to have all transaction parameters available at 'prep' time, saves two function pointers in struct dma_async_tx_descriptor, and reduces the number of indirect branches..

A consequence of moving this data to the 'prep' routine is that multi-source routines like async_xor need temporary storage to convert an array of linear addresses into an array of dma addresses. In order to keep the same stack footprint of the previous implementation the input array is reused as storage for the dma addresses. This requires that sizeof(dma_addr_t) be less than or equal to sizeof(void *). As a consequence CONFIG_DMADEVICES now depends on !CONFIG_HIGHMEM64G. It also requires that drivers be able to make descriptor resources available when the 'prep' routine is polled.

Signed-off-by: Dan Williams <dan.j.williams@xxxxxxxxxx>

```
crypto/async_tx/async_memcpy.c | 27 +++++-----
crypto/async_tx/async_memset.c | 20 +++----
crypto/async_tx/async_xor.c | 94 ++++++++-----
drivers/dma/Kconfig | 1
drivers/dma/dmaengine.c | 49 ++++++-----
drivers/dma/ioat_dma.c | 39 +++++-----
drivers/dma/iop-adma.c | 124 ++++++++-----
include/linux/dmaengine.h | 20 +++----
8 files changed, 178 insertions(+), 196 deletions(-)
```

```
diff --git a/crypto/async_tx/async_memcpy.c b/crypto/async_tx/async_memcpy.c
index e8c8956..faca0bc 100644
--- a/crypto/async_tx/async_memcpy.c
+++ b/crypto/async_tx/async_memcpy.c
@@ -48,26 +48,25 @@ async_memcpy(struct page *dest, struct page *src, unsigned int dest_offset,
{
struct dma_chan *chan = async_tx_find_channel(depend_tx, DMA_MEMCPY);
struct dma_device *device = chan ? chan->device : NULL;
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
- int int_en = cb_fn ? 1 : 0;
- struct dma_async_tx_descriptor *tx = device ?
- device->device_prep_dma_memcpy(chan, len,
- int_en) : NULL;
+ struct dma_async_tx_descriptor *tx = NULL;

- if (tx) { /* run the memcpy asynchronously */
- dma_addr_t addr;
+ if (device) {
+ dma_addr_t dma_dest, dma_src;

- pr_debug("%s: (async) len: %zu\n", __FUNCTION__, len);
+ dma_dest = dma_map_page(device->dev, dest, dest_offset, len,
+ DMA_FROM_DEVICE);

- addr = dma_map_page(device->dev, dest, dest_offset, len,
- DMA_FROM_DEVICE);
- tx->tx_set_dest(addr, tx, 0);
+ dma_src = dma_map_page(device->dev, src, src_offset, len,
+ DMA_TO_DEVICE);

- addr = dma_map_page(device->dev, src, src_offset, len,
- DMA_TO_DEVICE);
- tx->tx_set_src(addr, tx, 0);
+ tx = device->device_prep_dma_memcpy(chan, dma_dest, dma_src,
+ len, cb_fn != NULL);
+ }
```

```
+ if (tx) {
+ pr_debug("%s: (async) len: %zu\n", __FUNCTION__, len);
+ async_tx_submit(chan, tx, flags, depend_tx, cb_fn, cb_param);
- } else { /* run the memcpy synchronously */
+ } else {
void *dest_buf, *src_buf;
pr_debug("%s: (sync) len: %zu\n", __FUNCTION__, len);
```

```
diff --git a/crypto/async_tx/async_memset.c b/crypto/async_tx/async_memset.c
index 7609728..0c94851 100644
```

```
--- a/crypto/async_tx/async_memset.c
+++ b/crypto/async_tx/async_memset.c
@@ -48,20 +48,20 @@ async_memset(struct page *dest, int val, unsigned int offset,
{
struct dma_chan *chan = async_tx_find_channel(depend_tx, DMA_MEMSET);
struct dma_device *device = chan ? chan->device : NULL;
- int int_en = cb_fn ? 1 : 0;
- struct dma_async_tx_descriptor *tx = device ?
- device->device_prep_dma_memset(chan, val, len,
- int_en) : NULL;
+ struct dma_async_tx_descriptor *tx = NULL;

- if (tx) { /* run the memset asynchronously */
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
- dma_addr_t dma_addr;
+ if (device) {
+ dma_addr_t dma_dest;

- pr_debug("%s: (async) len: %zu\n", __FUNCTION__, len);
-
- dma_addr = dma_map_page(device->dev, dest, offset, len,
+ dma_dest = dma_map_page(device->dev, dest, offset, len,
DMA_FROM_DEVICE);
- tx->tx_set_dest(dma_addr, tx, 0);

+ tx = device->device_prep_dma_memset(chan, dma_dest, val, len,
+ cb_fn != NULL);
+ }
+
+ if (tx) {
+ pr_debug("%s: (async) len: %zu\n", __FUNCTION__, len);
+ async_tx_submit(chan, tx, flags, depend_tx, cb_fn, cb_param);
+ } else { /* run the memset synchronously */
+ void *dest_buf;
+ diff --git a/crypto/async_tx/async_xor.c b/crypto/async_tx/async_xor.c
+ index f332ddc..fbf113a 100644
+ --- a/crypto/async_tx/async_xor.c
+ +++ b/crypto/async_tx/async_xor.c
+ @@ -30,29 +30,46 @@
+ #include <linux/raid/xor.h>
+ #include <linux/async_tx.h>

- static void
- do_async_xor(struct dma_async_tx_descriptor *tx, struct dma_device *device,
+ static struct dma_async_tx_descriptor *
+ do_async_xor(struct dma_device *device,
+ struct dma_chan *chan, struct page *dest, struct page **src_list,
+ unsigned int offset, unsigned int src_cnt, size_t len,
+ enum async_tx_flags flags, struct dma_async_tx_descriptor *depend_tx,
+ dma_async_tx_callback cb_fn, void *cb_param)
+ {
+ dma_addr_t dma_addr;
+ dma_addr_t dma_dest;
+ dma_addr_t *dma_src = (dma_addr_t *) src_list;
+ struct dma_async_tx_descriptor *tx;
+ int i;

+ pr_debug("%s: len: %zu\n", __FUNCTION__, len);

- dma_addr = dma_map_page(device->dev, dest, offset, len,
+ dma_dest = dma_map_page(device->dev, dest, offset, len,
DMA_FROM_DEVICE);
- tx->tx_set_dest(dma_addr, tx, 0);

- for (i = 0; i < src_cnt; i++) {
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
- dma_addr = dma_map_page(device->dev, src_list[i],
- offset, len, DMA_TO_DEVICE);
- tx->tx_set_src(dma_addr, tx, i);
+ for (i = 0; i < src_cnt; i++)
+ dma_src[i] = dma_map_page(device->dev, src_list[i], offset,
+ len, DMA_TO_DEVICE);
+
+ /* Since we have clobbered the src_list we are committed
+ * to doing this asynchronously. Drivers force forward progress
+ * in case they can not provide a descriptor
+ */
+ tx = device->device_prep_dma_xor(chan, dma_dest, dma_src, src_cnt, len,
+ cb_fn != NULL);
+ if (!tx) {
+ if (depend_tx)
+ dma_wait_for_async_tx(depend_tx);
+
+ while (!tx)
+ tx = device->device_prep_dma_xor(chan, dma_dest,
+ dma_src, src_cnt, len,
+ cb_fn != NULL);
+ }

async_tx_submit(chan, tx, flags, depend_tx, cb_fn, cb_param);
+
+ return tx;
+ }

static void
@@ -114,7 +131,7 @@ async_xor(struct page *dest, struct page **src_list, unsigned int offset,
void *_cb_param;
unsigned long local_flags;
int xor_src_cnt;
- int i = 0, src_off = 0, int_en;
+ int i = 0, src_off = 0;

BUG_ON(src_cnt <= 1);

@@ -134,20 +151,11 @@ async_xor(struct page *dest, struct page **src_list, unsigned int offset,
_cb_param = cb_param;
}

- int_en = _cb_fn ? 1 : 0;
-
- tx = device->device_prep_dma_xor(
- chan, xor_src_cnt, len, int_en);
-
- if (tx) {
- do_async_xor(tx, device, chan, dest,
- &src_list[src_off], offset, xor_src_cnt, len,
- local_flags, depend_tx, _cb_fn,
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
- _cb_param);
- } else /* fall through */
- goto xor_sync;
+ tx = do_async_xor(device, chan, dest,
+ &src_list[src_off], offset,
+ xor_src_cnt, len, local_flags,
+ depend_tx, _cb_fn, _cb_param);
} else { /* run the xor synchronously */
-xor_sync:
/* in the sync case the dest is an implied source
* (assumes the dest is at the src_off index)
*/
@@ -250,23 +258,31 @@ async_xor_zero_sum(struct page *dest, struct page **src_list,
{
struct dma_chan *chan = async_tx_find_channel(depend_tx, DMA_ZERO_SUM);
struct dma_device *device = chan ? chan->device : NULL;
- int int_en = cb_fn ? 1 : 0;
- struct dma_async_tx_descriptor *tx = device ?
- device->device_prep_dma_zero_sum(chan, src_cnt, len, result,
- int_en) : NULL;
- int i;
+ struct dma_async_tx_descriptor *tx = NULL;

BUG_ON(src_cnt <= 1);

- if (tx) {
- dma_addr_t dma_addr;
+ if (device) {
+ dma_addr_t *dma_src = (dma_addr_t *) src_list;
+ int i;

pr_debug("%s: (async) len: %zu\n", __FUNCTION__, len);

- for (i = 0; i < src_cnt; i++) {
- dma_addr = dma_map_page(device->dev, src_list[i],
- offset, len, DMA_TO_DEVICE);
- tx->tx_set_src(dma_addr, tx, i);
+ for (i = 0; i < src_cnt; i++)
+ dma_src[i] = dma_map_page(device->dev, src_list[i],
+ offset, len, DMA_TO_DEVICE);
+
+ tx = device->device_prep_dma_zero_sum(chan, dma_src, src_cnt,
+ len, result,
+ cb_fn != NULL);
+ if (!tx) {
+ if (depend_tx)
+ dma_wait_for_async_tx(depend_tx);
+
+ while (!tx)
+ tx = device->device_prep_dma_zero_sum(chan,
+ dma_src, src_cnt, len, result,
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
+ cb_fn != NULL);  
}
```

```
async_tx_submit(chan, tx, flags, depend_tx, cb_fn, cb_param);  
@@ -301,6 +317,16 @@ EXPORT_SYMBOL_GPL(async_xor_zero_sum);
```

```
static int __init async_xor_init(void)  
{  
+ #ifdef CONFIG_DMA_ENGINE  
+ /* To conserve stack space the input src_list (array of page pointers)  
+ * is reused to hold the array of dma addresses passed to the driver.  
+ * This conversion is only possible when dma_addr_t is less than the  
+ * the size of a pointer. HIGHMEM64G is known to violate this  
+ * assumption.  
+ */  
+ BUILD_BUG_ON(sizeof(dma_addr_t) > sizeof(struct page *));  
+ #endif  
+  
return 0;  
}
```

```
diff --git a/drivers/dma/Kconfig b/drivers/dma/Kconfig  
index c46b7c2..a703def 100644
```

```
--- a/drivers/dma/Kconfig
```

```
+++ b/drivers/dma/Kconfig
```

```
@@ -5,6 +5,7 @@
```

```
menuconfig DMADEVICES
```

```
bool "DMA Engine support"
```

```
depends on (PCI && X86) || ARCH_IOP32X || ARCH_IOP33X || ARCH_IOP13XX
```

```
+ depends on !HIGHMEM64G
```

```
help
```

DMA engines can do asynchronous data transfers without involving the host CPU. Currently, this framework can be

```
diff --git a/drivers/dma/dmaengine.c b/drivers/dma/dmaengine.c
```

```
index ec7e871..1adb9de 100644
```

```
--- a/drivers/dma/dmaengine.c
```

```
+++ b/drivers/dma/dmaengine.c
```

```
@@ -476,20 +476,22 @@ dma_async_memcpy_buf_to_buf(struct dma_chan *chan, void *dest,
```

```
{
```

```
struct dma_device *dev = chan->device;
```

```
struct dma_async_tx_descriptor *tx;
```

```
- dma_addr_t addr;
```

```
+ dma_addr_t dma_dest, dma_src;
```

```
dma_cookie_t cookie;
```

```
int cpu;
```

```
- tx = dev->device_prep_dma_memcpy(chan, len, 0);
```

```
- if (!tx)
```

```
+ dma_src = dma_map_single(dev->dev, src, len, DMA_TO_DEVICE);
```

```
+ dma_dest = dma_map_single(dev->dev, dest, len, DMA_FROM_DEVICE);
```

```
+ tx = dev->device_prep_dma_memcpy(chan, dma_dest, dma_src, len, 0);
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
+
+ if (!tx) {
+ dma_unmap_single(dev->dev, dma_src, len, DMA_TO_DEVICE);
+ dma_unmap_single(dev->dev, dma_dest, len, DMA_FROM_DEVICE);
return -ENOMEM;
+ }

tx->ack = 1;
tx->callback = NULL;
- addr = dma_map_single(dev->dev, src, len, DMA_TO_DEVICE);
- tx->tx_set_src(addr, tx, 0);
- addr = dma_map_single(dev->dev, dest, len, DMA_FROM_DEVICE);
- tx->tx_set_dest(addr, tx, 0);
cookie = tx->tx_submit(tx);

cpu = get_cpu();
@@ -520,20 +522,22 @@ dma_async_memcpy_buf_to_pg(struct dma_chan *chan, struct page *page,
{
struct dma_device *dev = chan->device;
struct dma_async_tx_descriptor *tx;
- dma_addr_t addr;
+ dma_addr_t dma_dest, dma_src;
dma_cookie_t cookie;
int cpu;

- tx = dev->device_prep_dma_memcpy(chan, len, 0);
- if (!tx)
+ dma_src = dma_map_single(dev->dev, kdata, len, DMA_TO_DEVICE);
+ dma_dest = dma_map_page(dev->dev, page, offset, len, DMA_FROM_DEVICE);
+ tx = dev->device_prep_dma_memcpy(chan, dma_dest, dma_src, len, 0);
+
+ if (!tx) {
+ dma_unmap_single(dev->dev, dma_src, len, DMA_TO_DEVICE);
+ dma_unmap_page(dev->dev, dma_dest, len, DMA_FROM_DEVICE);
return -ENOMEM;
+ }

tx->ack = 1;
tx->callback = NULL;
- addr = dma_map_single(dev->dev, kdata, len, DMA_TO_DEVICE);
- tx->tx_set_src(addr, tx, 0);
- addr = dma_map_page(dev->dev, page, offset, len, DMA_FROM_DEVICE);
- tx->tx_set_dest(addr, tx, 0);
cookie = tx->tx_submit(tx);

cpu = get_cpu();
@@ -566,20 +570,23 @@ dma_async_memcpy_pg_to_pg(struct dma_chan *chan, struct page *dest_pg,
{
struct dma_device *dev = chan->device;
struct dma_async_tx_descriptor *tx;
- dma_addr_t addr;
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
+ dma_addr_t dma_dest, dma_src;
dma_cookie_t cookie;
int cpu;

- tx = dev->device_prep_dma_memcpy(chan, len, 0);
- if (!tx)
+ dma_src = dma_map_page(dev->dev, src_pg, src_off, len, DMA_TO_DEVICE);
+ dma_dest = dma_map_page(dev->dev, dest_pg, dest_off, len,
+ DMA_FROM_DEVICE);
+ tx = dev->device_prep_dma_memcpy(chan, dma_dest, dma_src, len, 0);
+
+ if (!tx) {
+ dma_unmap_page(dev->dev, dma_src, len, DMA_TO_DEVICE);
+ dma_unmap_page(dev->dev, dma_dest, len, DMA_FROM_DEVICE);
return -ENOMEM;
+ }

tx->ack = 1;
tx->callback = NULL;
- addr = dma_map_page(dev->dev, src_pg, src_off, len, DMA_TO_DEVICE);
- tx->tx_set_src(addr, tx, 0);
- addr = dma_map_page(dev->dev, dest_pg, dest_off, len, DMA_FROM_DEVICE);
- tx->tx_set_dest(addr, tx, 0);
cookie = tx->tx_submit(tx);

cpu = get_cpu();
diff --git a/drivers/dma/ioat_dma.c b/drivers/dma/ioat_dma.c
index 45e7b46..5bcfc55 100644
--- a/drivers/dma/ioat_dma.c
+++ b/drivers/dma/ioat_dma.c
@@ -159,20 +159,6 @@ static int ioat_dma_enumerate_channels(struct ioatdma_device *device)
return device->common.chancnt;
}

-static void ioat_set_src(dma_addr_t addr,
- struct dma_async_tx_descriptor *tx,
- int index)
-{
- tx_to_ioat_desc(tx)->src = addr;
-}
-
-static void ioat_set_dest(dma_addr_t addr,
- struct dma_async_tx_descriptor *tx,
- int index)
-{
- tx_to_ioat_desc(tx)->dst = addr;
-}
-
/**
 * ioat_dma_memcpy_issue_pending - push potentially unrecognized appended
 * descriptors to hw
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
@@ -415,8 +401,6 @@ static struct ioat_desc_sw *ioat_dma_alloc_descriptor(
memset(desc, 0, sizeof(*desc));
dma_async_tx_descriptor_init(&desc_sw->async_tx, &ioat_chan->common);
- desc_sw->async_tx.tx_set_src = ioat_set_src;
- desc_sw->async_tx.tx_set_dest = ioat_set_dest;
switch (ioat_chan->device->version) {
case IOAT_VER_1_2:
desc_sw->async_tx.tx_submit = ioat1_tx_submit;
@@ -714,6 +698,8 @@ static struct ioat_desc_sw *ioat_dma_get_next_descriptor(

static struct dma_async_tx_descriptor *ioat1_dma_prep_memcpy(
struct dma_chan *chan,
+ dma_addr_t dma_dest,
+ dma_addr_t dma_src,
size_t len,
int int_en)
{
@@ -726,6 +712,8 @@ static struct dma_async_tx_descriptor *ioat1_dma_prep_memcpy(

if (new) {
new->len = len;
+ new->dst = dma_dest;
+ new->src = dma_src;
return &new->async_tx;
} else
return NULL;
@@ -733,6 +721,8 @@ static struct dma_async_tx_descriptor *ioat1_dma_prep_memcpy(

static struct dma_async_tx_descriptor *ioat2_dma_prep_memcpy(
struct dma_chan *chan,
+ dma_addr_t dma_dest,
+ dma_addr_t dma_src,
size_t len,
int int_en)
{
@@ -749,6 +739,8 @@ static struct dma_async_tx_descriptor *ioat2_dma_prep_memcpy(

if (new) {
new->len = len;
+ new->dst = dma_dest;
+ new->src = dma_src;
return &new->async_tx;
} else
return NULL;
@@ -1045,7 +1037,7 @@ static int ioat_dma_self_test(struct ioatdma_device *device)
u8 *dest;
struct dma_chan *dma_chan;
struct dma_async_tx_descriptor *tx;
- dma_addr_t addr;
+ dma_addr_t dma_dest, dma_src;
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
dma_cookie_t cookie;
int err = 0;

@@ -1073,7 +1065,12 @@ static int ioat_dma_self_test(struct ioatdma_device *device)
goto out;
}

- tx = device->common.device_prep_dma_memcpy(dma_chan, IOAT_TEST_SIZE, 0);
+ dma_src = dma_map_single(dma_chan->device->dev, src, IOAT_TEST_SIZE,
+ DMA_TO_DEVICE);
+ dma_dest = dma_map_single(dma_chan->device->dev, dest, IOAT_TEST_SIZE,
+ DMA_FROM_DEVICE);
+ tx = device->common.device_prep_dma_memcpy(dma_chan, dma_dest, dma_src,
+ IOAT_TEST_SIZE, 0);
if (!tx) {
dev_err(&device->pdev->dev,
"Self-test prep failed, disabling\n");
@@ -1082,12 +1079,6 @@ static int ioat_dma_self_test(struct ioatdma_device *device)
}

async_tx_ack(tx);
- addr = dma_map_single(dma_chan->device->dev, src, IOAT_TEST_SIZE,
- DMA_TO_DEVICE);
- tx->tx_set_src(addr, tx, 0);
- addr = dma_map_single(dma_chan->device->dev, dest, IOAT_TEST_SIZE,
- DMA_FROM_DEVICE);
- tx->tx_set_dest(addr, tx, 0);
tx->callback = ioat_dma_test_callback;
tx->callback_param = (void *)0x8086;
cookie = tx->tx_submit(tx);
diff --git a/drivers/dma/iop-adma.c b/drivers/dma/iop-adma.c
index b011b5a..eda841c 100644
--- a/drivers/dma/iop-adma.c
+++ b/drivers/dma/iop-adma.c
@@ -443,17 +443,6 @@ iop_adma_tx_submit(struct dma_async_tx_descriptor *tx)
return cookie;
}

-static void
-iop_adma_set_dest(dma_addr_t addr, struct dma_async_tx_descriptor *tx,
- int index)
-{
- struct iop_adma_desc_slot *sw_desc = tx_to_iop_adma_slot(tx);
- struct iop_adma_chan *iop_chan = to_iop_adma_chan(tx->chan);
-
- /* to do: support transfers lengths > IOP_ADMA_MAX_BYTE_COUNT */
- iop_desc_set_dest_addr(sw_desc->group_head, iop_chan, addr);
-}

static void iop_chan_start_null_memcpy(struct iop_adma_chan *iop_chan);
static void iop_chan_start_null_xor(struct iop_adma_chan *iop_chan);
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
@@ -486,7 +475,6 @@ static int iop_adma_alloc_chan_resources(struct dma_chan *chan)

dma_async_tx_descriptor_init(&slot->async_tx, chan);
slot->async_tx.tx_submit = iop_adma_tx_submit;
- slot->async_tx.tx_set_dest = iop_adma_set_dest;
INIT_LIST_HEAD(&slot->chain_node);
INIT_LIST_HEAD(&slot->slot_node);
INIT_LIST_HEAD(&slot->async_tx.tx_list);
@@ -547,18 +535,9 @@ iop_adma_prep_dma_interrupt(struct dma_chan *chan)
return sw_desc ? &sw_desc->async_tx : NULL;
}

-static void
-iop_adma_memcpy_set_src(dma_addr_t addr, struct dma_async_tx_descriptor *tx,
- int index)
-{
- struct iop_adma_desc_slot *sw_desc = tx_to_iop_adma_slot(tx);
- struct iop_adma_desc_slot *grp_start = sw_desc->group_head;
-
- iop_desc_set_memcpy_src_addr(grp_start, addr);
-}
-
static struct dma_async_tx_descriptor *
-iop_adma_prep_dma_memcpy(struct dma_chan *chan, size_t len, int int_en)
+iop_adma_prep_dma_memcpy(struct dma_chan *chan, dma_addr_t dma_dest,
+ dma_addr_t dma_src, size_t len, int int_en)
{
struct iop_adma_chan *iop_chan = to_iop_adma_chan(chan);
struct iop_adma_desc_slot *sw_desc, *grp_start;
@@ -578,9 +557,10 @@ iop_adma_prep_dma_memcpy(struct dma_chan *chan, size_t len, int int_en)
grp_start = sw_desc->group_head;
iop_desc_init_memcpy(grp_start, int_en);
iop_desc_set_byte_count(grp_start, iop_chan, len);
+ iop_desc_set_dest_addr(grp_start, iop_chan, dma_dest);
+ iop_desc_set_memcpy_src_addr(grp_start, dma_src);
sw_desc->unmap_src_cnt = 1;
sw_desc->unmap_len = len;
- sw_desc->async_tx.tx_set_src = iop_adma_memcpy_set_src;
}
spin_unlock_bh(&iop_chan->lock);

@@ -588,8 +568,8 @@ iop_adma_prep_dma_memcpy(struct dma_chan *chan, size_t len, int int_en)
}

static struct dma_async_tx_descriptor *
-iop_adma_prep_dma_memset(struct dma_chan *chan, int value, size_t len,
- int int_en)
+iop_adma_prep_dma_memset(struct dma_chan *chan, dma_addr_t dma_dest,
+ int value, size_t len, int int_en)
{
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
struct iop_adma_chan *iop_chan = to_iop_adma_chan(chan);
struct iop_adma_desc_slot *sw_desc, *grp_start;
@@ -610,6 +590,7 @@ iop_adma_prep_dma_memset(struct dma_chan *chan, int value, size_t len,
iop_desc_init_memset(grp_start, int_en);
iop_desc_set_byte_count(grp_start, iop_chan, len);
iop_desc_set_block_fill_val(grp_start, value);
+ iop_desc_set_dest_addr(grp_start, iop_chan, dma_dest);
sw_desc->unmap_src_cnt = 1;
sw_desc->unmap_len = len;
}
@@ -618,19 +599,10 @@ iop_adma_prep_dma_memset(struct dma_chan *chan, int value, size_t len,
return sw_desc ? &sw_desc->async_tx : NULL;
}

-static void
-iop_adma_xor_set_src(dma_addr_t addr, struct dma_async_tx_descriptor *tx,
- int index)
-{
- struct iop_adma_desc_slot *sw_desc = tx_to_iop_adma_slot(tx);
- struct iop_adma_desc_slot *grp_start = sw_desc->group_head;
-
- iop_desc_set_xor_src_addr(grp_start, index, addr);
-}
-
static struct dma_async_tx_descriptor *
iop_adma_prep_dma_xor(struct dma_chan *chan, unsigned int src_cnt, size_t len,
- int int_en)
+iop_adma_prep_dma_xor(struct dma_chan *chan, dma_addr_t dma_dest,
+ dma_addr_t *dma_src, unsigned int src_cnt, size_t len,
+ int int_en)
{
struct iop_adma_chan *iop_chan = to_iop_adma_chan(chan);
struct iop_adma_desc_slot *sw_desc, *grp_start;
@@ -651,29 +623,22 @@ iop_adma_prep_dma_xor(struct dma_chan *chan, unsigned int src_cnt, size_t len,
grp_start = sw_desc->group_head;
iop_desc_init_xor(grp_start, src_cnt, int_en);
iop_desc_set_byte_count(grp_start, iop_chan, len);
+ iop_desc_set_dest_addr(grp_start, iop_chan, dma_dest);
sw_desc->unmap_src_cnt = src_cnt;
sw_desc->unmap_len = len;
- sw_desc->async_tx.tx_set_src = iop_adma_xor_set_src;
+ while (src_cnt--)
+ iop_desc_set_xor_src_addr(grp_start, src_cnt,
+ dma_src[src_cnt]);
}
spin_unlock_bh(&iop_chan->lock);

return sw_desc ? &sw_desc->async_tx : NULL;
}

-static void
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
-iop_adma_xor_zero_sum_set_src(dma_addr_t addr,
- struct dma_async_tx_descriptor *tx,
- int index)
- {
- struct iop_adma_desc_slot *sw_desc = tx_to_iop_adma_slot(tx);
- struct iop_adma_desc_slot *grp_start = sw_desc->group_head;
-
- iop_desc_set_zero_sum_src_addr(grp_start, index, addr);
- }
-
static struct dma_async_tx_descriptor *
-iop_adma_prep_dma_zero_sum(struct dma_chan *chan, unsigned int src_cnt,
- size_t len, u32 *result, int int_en)
+iop_adma_prep_dma_zero_sum(struct dma_chan *chan, dma_addr_t *dma_src,
+ unsigned int src_cnt, size_t len, u32 *result,
+ int int_en)
{
struct iop_adma_chan *iop_chan = to_iop_adma_chan(chan);
struct iop_adma_desc_slot *sw_desc, *grp_start;
@@ -697,7 +662,9 @@ iop_adma_prep_dma_zero_sum(struct dma_chan *chan, unsigned int src_cnt,
__FUNCTION__, grp_start->xor_check_result);
sw_desc->unmap_src_cnt = src_cnt;
sw_desc->unmap_len = len;
- sw_desc->async_tx.tx_set_src = iop_adma_xor_zero_sum_set_src;
+ while (src_cnt-->0)
+ iop_desc_set_zero_sum_src_addr(grp_start, src_cnt,
+ dma_src[src_cnt]);
}
spin_unlock_bh(&iop_chan->lock);

@@ -882,13 +849,12 @@ static int __devinit iop_adma_memcpy_self_test(struct iop_adma_device
*device)
goto out;
}

- tx = iop_adma_prep_dma_memcpy(dma_chan, IOP_ADMA_TEST_SIZE, 1);
dest_dma = dma_map_single(dma_chan->device->dev, dest,
IOP_ADMA_TEST_SIZE, DMA_FROM_DEVICE);
- iop_adma_set_dest(dest_dma, tx, 0);
src_dma = dma_map_single(dma_chan->device->dev, src,
IOP_ADMA_TEST_SIZE, DMA_TO_DEVICE);
- iop_adma_memcpy_set_src(src_dma, tx, 0);
+ tx = iop_adma_prep_dma_memcpy(dma_chan, dest_dma, src_dma,
+ IOP_ADMA_TEST_SIZE, 1);

cookie = iop_adma_tx_submit(tx);
iop_adma_issue_pending(dma_chan);
@@ -929,6 +895,7 @@ iop_adma_xor_zero_sum_self_test(struct iop_adma_device *device)
struct page *dest;
struct page *xor_srcs[IOP_ADMA_NUM_SRC_TEST];
struct page *zero_sum_srcs[IOP_ADMA_NUM_SRC_TEST + 1];
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
+ dma_addr_t dma_srcs[IOP_ADMA_NUM_SRC_TEST + 1];
dma_addr_t dma_addr, dest_dma;
struct dma_async_tx_descriptor *tx;
struct dma_chan *dma_chan;
@@ -981,17 +948,13 @@ iop_adma_xor_zero_sum_self_test(struct iop_adma_device *device)
}

/* test xor */
- tx = iop_adma_prep_dma_xor(dma_chan, IOP_ADMA_NUM_SRC_TEST,
- PAGE_SIZE, 1);
dest_dma = dma_map_page(dma_chan->device->dev, dest, 0,
PAGE_SIZE, DMA_FROM_DEVICE);
- iop_adma_set_dest(dest_dma, tx, 0);
-
- for (i = 0; i < IOP_ADMA_NUM_SRC_TEST; i++) {
- dma_addr = dma_map_page(dma_chan->device->dev, xor_srcs[i], 0,
- PAGE_SIZE, DMA_TO_DEVICE);
- iop_adma_xor_set_src(dma_addr, tx, i);
- }
+ for (i = 0; i < IOP_ADMA_NUM_SRC_TEST; i++)
+ dma_srcs[i] = dma_map_page(dma_chan->device->dev, xor_srcs[i],
+ 0, PAGE_SIZE, DMA_TO_DEVICE);
+ tx = iop_adma_prep_dma_xor(dma_chan, dest_dma, dma_srcs,
+ IOP_ADMA_NUM_SRC_TEST, PAGE_SIZE, 1);

cookie = iop_adma_tx_submit(tx);
iop_adma_issue_pending(dma_chan);
@@ -1032,13 +995,13 @@ iop_adma_xor_zero_sum_self_test(struct iop_adma_device *device)

zero_sum_result = 1;

- tx = iop_adma_prep_dma_zero_sum(dma_chan, IOP_ADMA_NUM_SRC_TEST + 1,
- PAGE_SIZE, &zero_sum_result, 1);
- for (i = 0; i < IOP_ADMA_NUM_SRC_TEST + 1; i++) {
- dma_addr = dma_map_page(dma_chan->device->dev, zero_sum_srcs[i],
- 0, PAGE_SIZE, DMA_TO_DEVICE);
- iop_adma_xor_zero_sum_set_src(dma_addr, tx, i);
- }
+ for (i = 0; i < IOP_ADMA_NUM_SRC_TEST + 1; i++)
+ dma_srcs[i] = dma_map_page(dma_chan->device->dev,
+ zero_sum_srcs[i], 0, PAGE_SIZE,
+ DMA_TO_DEVICE);
+ tx = iop_adma_prep_dma_zero_sum(dma_chan, dma_srcs,
+ IOP_ADMA_NUM_SRC_TEST + 1, PAGE_SIZE,
+ &zero_sum_result, 1);

cookie = iop_adma_tx_submit(tx);
iop_adma_issue_pending(dma_chan);
@@ -1060,10 +1023,9 @@ iop_adma_xor_zero_sum_self_test(struct iop_adma_device *device)
}
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
/* test memset */
- tx = iop_adma_prep_dma_memset(dma_chan, 0, PAGE_SIZE, 1);
dma_addr = dma_map_page(dma_chan->device->dev, dest, 0,
PAGE_SIZE, DMA_FROM_DEVICE);
- iop_adma_set_dest(dma_addr, tx, 0);
+ tx = iop_adma_prep_dma_memset(dma_chan, dma_addr, 0, PAGE_SIZE, 1);

cookie = iop_adma_tx_submit(tx);
iop_adma_issue_pending(dma_chan);
@@ -1089,13 +1051,13 @@ iop_adma_xor_zero_sum_self_test(struct iop_adma_device *device)

/* test for non-zero parity sum */
zero_sum_result = 0;
- tx = iop_adma_prep_dma_zero_sum(dma_chan, IOP_ADMA_NUM_SRC_TEST + 1,
- PAGE_SIZE, &zero_sum_result, 1);
- for (i = 0; i < IOP_ADMA_NUM_SRC_TEST + 1; i++) {
- dma_addr = dma_map_page(dma_chan->device->dev, zero_sum_srcs[i],
- 0, PAGE_SIZE, DMA_TO_DEVICE);
- iop_adma_xor_zero_sum_set_src(dma_addr, tx, i);
- }
+ for (i = 0; i < IOP_ADMA_NUM_SRC_TEST + 1; i++)
+ dma_srcs[i] = dma_map_page(dma_chan->device->dev,
+ zero_sum_srcs[i], 0, PAGE_SIZE,
+ DMA_TO_DEVICE);
+ tx = iop_adma_prep_dma_zero_sum(dma_chan, dma_srcs,
+ IOP_ADMA_NUM_SRC_TEST + 1, PAGE_SIZE,
+ &zero_sum_result, 1);

cookie = iop_adma_tx_submit(tx);
iop_adma_issue_pending(dma_chan);
diff --git a/include/linux/dmaengine.h b/include/linux/dmaengine.h
index 55c9a69..e38af8b 100644
--- a/include/linux/dmaengine.h
+++ b/include/linux/dmaengine.h
@@ -209,8 +209,6 @@ typedef void (*dma_async_tx_callback)(void *dma_async_param);
* descriptors
* @chan: target channel for this operation
* @tx_submit: set the prepared descriptor(s) to be executed by the engine
- * @tx_set_dest: set a destination address in a hardware descriptor
- * @tx_set_src: set a source address in a hardware descriptor
* @callback: routine to call after this operation is complete
* @callback_param: general parameter to pass to the callback routine
* ---async_tx api specific fields---
@@ -227,10 +225,6 @@ struct dma_async_tx_descriptor {
struct list_head tx_list;
struct dma_chan *chan;
dma_cookie_t (*tx_submit)(struct dma_async_tx_descriptor *tx);
- void (*tx_set_dest)(dma_addr_t addr,
- struct dma_async_tx_descriptor *tx, int index);
- void (*tx_set_src)(dma_addr_t addr,
- struct dma_async_tx_descriptor *tx, int index);
```

[PATCH 2/4] async_tx: kill tx_set_src and tx_set_dest methods

```
dma_async_tx_callback callback;
void *callback_param;
struct list_head depend_list;
@@ -279,15 +273,17 @@ struct dma_device {
void (*device_free_chan_resources)(struct dma_chan *chan);

struct dma_async_tx_descriptor *(*device_prep_dma_memcpy)(
- struct dma_chan *chan, size_t len, int int_en);
+ struct dma_chan *chan, dma_addr_t dest, dma_addr_t src,
+ size_t len, int int_en);
struct dma_async_tx_descriptor *(*device_prep_dma_xor)(
- struct dma_chan *chan, unsigned int src_cnt, size_t len,
- int int_en);
+ struct dma_chan *chan, dma_addr_t dest, dma_addr_t *src,
+ unsigned int src_cnt, size_t len, int int_en);
struct dma_async_tx_descriptor *(*device_prep_dma_zero_sum)(
- struct dma_chan *chan, unsigned int src_cnt, size_t len,
- u32 *result, int int_en);
+ struct dma_chan *chan, dma_addr_t *src, unsigned int src_cnt,
+ size_t len, u32 *result, int int_en);
struct dma_async_tx_descriptor *(*device_prep_dma_memset)(
- struct dma_chan *chan, int value, size_t len, int int_en);
+ struct dma_chan *chan, dma_addr_t dest, int value, size_t len,
+ int int_en);
struct dma_async_tx_descriptor *(*device_prep_dma_interrupt)(
struct dma_chan *chan);
```

--

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/>