author	Dan Williams <dan.j.williams@intel.com>	2022-11-03 17:30:36 -0700
committer	Greg Kroah-Hartman <gregkh@linuxfoundation.org>	2022-11-10 18:17:32 +0100
commit	f43b6bfdbab78606735ba81185cf0602b81e40b6 (patch)	
tree	d00fb52611d9db6098e81a00230f890388c00d9e	
parent	f1ad8f211c67961a00fb3a24c7d305d76d2b16c0 (diff)	
download	linux-f43b6bfdbab78606735ba81185cf0602b81e40b6.tar.gz	

cxl/pmem: Fix cxl_pmem_region and cxl_memdev leak

```
commit 4d07ae22e79ebc2d7528bbc69daa53b86981cb3a upstream.

When a cxl_nvdimm object goes through a ->remove() event (device
physically removed, nvdimm-bridge disabled, or nvdimm device disabled),
then any associated regions must also be disabled. As highlighted by the
cxl-create-region.sh test [1], a single device may host multiple
regions, but the driver was only tracking one region at a time. This
leads to a situation where only the last enabled region per nvdimm
device is cleaned up properly. Other regions are leaked, and this also
causes cxl_memdev reference leaks.

Fix the tracking by allowing cxl_nvdimm objects to track multiple region
associations.

Cc: <stable@vger.kernel.org>
Link: https://github.com/pmem/ndctl/blob/main/test/cxl-create-region.sh [1]
Reported-by: Vishal Verma <vishal.l.verma@intel.com>
Fixes: 04ad63f086d1 ("cxl/region: Introduce cxl_pmem_region objects")
Reviewed-by: Dave Jiang <dave.jiang@intel.com>
Reviewed-by: Vishal Verma <vishal.l.verma@intel.com>
Link: https://lore.kernel.org/r/166752183647.947915.2045230911503793901.stgit@dwillia2-xfh.jf.intel.com
Signed-off-by: Dan Williams <dan.j.williams@intel.com>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

Diffstat

-rw-r--r--	drivers/cxl/core/pmem.c	2
-rw-r--r--	drivers/cxl/cxl.h	2
-rw-r--r--	drivers/cxl/pmem.c	101

3 files changed, 68 insertions, 37 deletions

```
diff --git a/drivers/cxl/core/pmem.c b/drivers/cxl/core/pmem.c
index 1d12a8206444ef..36aa5070d90241 100644
--- a/drivers/cxl/core/pmem.c
+++ b/drivers/cxl/core/pmem.c
@@ -188,6 +188,7 @@ static void cxl_nvdimm_release(struct device *dev)
 {
        struct cxl_nvdimm *cxl_nvd = to_cxl_nvdimm(dev);

+       xa_destroy(&cxl_nvd->pmem_regions);
        kfree(cxl_nvd);
 }

@@ -230,6 +231,7 @@ static struct cxl_nvdimm *cxl_nvdimm_alloc(struct cxl_memdev *cxlmd)
```

```
        dev = &cxl_nvd->dev;
        cxl_nvd->cxlmd = cxlmd;
+       xa_init(&cxl_nvd->pmem_regions);
        device_initialize(dev);
        lockdep_set_class(&dev->mutex, &cxl_nvdimm_key);
        device_set_pm_not_required(dev);

diff --git a/drivers/cxl/cxl.h b/drivers/cxl/cxl.h
index f680450f0b16c8..1164ad49f3d3a7 100644
--- a/drivers/cxl/cxl.h
+++ b/drivers/cxl/cxl.h
@@ -423,7 +423,7 @@ struct cxl_nvdimm {
        struct device dev;
        struct cxl_memdev *cxlmd;
        struct cxl_nvdimm_bridge *bridge;
-       struct cxl_pmem_region *region;
+       struct xarray pmem_regions;
 };

 struct cxl_pmem_region_mapping {

diff --git a/drivers/cxl/pmem.c b/drivers/cxl/pmem.c
index 7dc0a2fa1a6b61..faade12279f02c 100644
--- a/drivers/cxl/pmem.c
+++ b/drivers/cxl/pmem.c
@@ -30,17 +30,20 @@ static void unregister_nvdimm(void *nvdimm)
        struct cxl_nvdimm *cxl_nvd = nvdimm_provider_data(nvdimm);
        struct cxl_nvdimm_bridge *cxl_nvb = cxl_nvd->bridge;
        struct cxl_pmem_region *cxlr_pmem;
+       unsigned long index;

        device_lock(&cxl_nvb->dev);
-       cxlr_pmem = cxl_nvd->region;
        dev_set_drvdata(&cxl_nvd->dev, NULL);
-       cxl_nvd->region = NULL;
-       device_unlock(&cxl_nvb->dev);
+       xa_for_each(&cxl_nvd->pmem_regions, index, cxlr_pmem) {
+               get_device(&cxlr_pmem->dev);
+               device_unlock(&cxl_nvb->dev);

-       if (cxlr_pmem) {
                device_release_driver(&cxlr_pmem->dev);
                put_device(&cxlr_pmem->dev);
+
+               device_lock(&cxl_nvb->dev);
        }
+       device_unlock(&cxl_nvb->dev);

        nvdimm_delete(nvdimm);
        cxl_nvd->bridge = NULL;
@@ -366,25 +369,49 @@ static int match_cxl_nvdimm(struct device *dev, void *data)

 static void unregister_nvdimm_region(void *nd_region)
 {
-       struct cxl_nvdimm_bridge *cxl_nvb;
-       struct cxl_pmem_region *cxlr_pmem;
+       nvdimm_region_delete(nd_region);
+}
+
+static int cxl_nvdimm_add_region(struct cxl_nvdimm *cxl_nvd,
+                                struct cxl_pmem_region *cxlr_pmem)
+{
+       int rc;
+
+       rc = xa_insert(&cxl_nvd->pmem_regions, (unsigned long)cxlr_pmem,
+                      cxlr_pmem, GFP_KERNEL);
```

```
+       if (rc)
+               return rc;
+
+       get_device(&cxlr_pmem->dev);
+       return 0;
+}
+
+static void cxl_nvdimm_del_region(struct cxl_nvdimm *cxl_nvd,
+                                 struct cxl_pmem_region *cxlr_pmem)
+{
+       /*
+        * It is possible this is called without a corresponding
+        * cxl_nvdimm_add_region for @cxlr_pmem
+        */
+       cxlr_pmem = xa_erase(&cxl_nvd->pmem_regions, (unsigned long)cxlr_pmem);
+       if (cxlr_pmem)
+               put_device(&cxlr_pmem->dev);
+}
+
+static void release_mappings(void *data)
+{
        int i;
+       struct cxl_pmem_region *cxlr_pmem = data;
+       struct cxl_nvdimm_bridge *cxl_nvb = cxlr_pmem->bridge;

-       cxlr_pmem = nd_region_provider_data(nd_region);
-       cxl_nvb = cxlr_pmem->bridge;
        device_lock(&cxl_nvb->dev);
        for (i = 0; i < cxlr_pmem->nr_mappings; i++) {
                struct cxl_pmem_region_mapping *m = &cxlr_pmem->mapping[i];
                struct cxl_nvdimm *cxl_nvd = m->cxl_nvd;

-               if (cxl_nvd->region) {
-                       put_device(&cxlr_pmem->dev);
-                       cxl_nvd->region = NULL;
-               }
+               cxl_nvdimm_del_region(cxl_nvd, cxlr_pmem);
        }
        device_unlock(&cxl_nvb->dev);
-
-       nvdimm_region_delete(nd_region);
 }

 static void cxlr_pmem_remove_resource(void *res)
@@ -422,7 +449,7 @@ static int cxl_pmem_region_probe(struct device *dev)
        if (!cxl_nvb->nvdimm_bus) {
                dev_dbg(dev, "nvdimm bus not found\n");
                rc = -ENXIO;
-               goto err;
+               goto out_nvb;
        }

        memset(&mappings, 0, sizeof(mappings));
@@ -431,7 +458,7 @@ static int cxl_pmem_region_probe(struct device *dev)
        res = devm_kzalloc(dev, sizeof(*res), GFP_KERNEL);
        if (!res) {
                rc = -ENOMEM;
-               goto err;
+               goto out_nvb;
        }

        res->name = "Persistent Memory";
@@ -442,11 +469,11 @@ static int cxl_pmem_region_probe(struct device *dev)

        rc = insert_resource(&iomem_resource, res);
```

```
        if (rc)
-               goto err;
+               goto out_nvb;

        rc = devm_add_action_or_reset(dev, cxlr_pmem_remove_resource, res);
        if (rc)
-               goto err;
+               goto out_nvb;

        ndr_desc.res = res;
        ndr_desc.provider_data = cxlr_pmem;
@@ -462,7 +489,7 @@ static int cxl_pmem_region_probe(struct device *dev)
        nd_set = devm_kzalloc(dev, sizeof(*nd_set), GFP_KERNEL);
        if (!nd_set) {
                rc = -ENOMEM;
-               goto err;
+               goto out_nvb;
        }

        ndr_desc.memregion = cxlr->id;
@@ -472,9 +499,13 @@ static int cxl_pmem_region_probe(struct device *dev)
        info = kmalloc_array(cxlr_pmem->nr_mappings, sizeof(*info), GFP_KERNEL);
        if (!info) {
                rc = -ENOMEM;
-               goto err;
+               goto out_nvb;
        }

+       rc = devm_add_action_or_reset(dev, release_mappings, cxlr_pmem);
+       if (rc)
+               goto out_nvd;
+
        for (i = 0; i < cxlr_pmem->nr_mappings; i++) {
                struct cxl_pmem_region_mapping *m = &cxlr_pmem->mapping[i];
                struct cxl_memdev *cxlmd = m->cxlmd;
@@ -486,7 +517,7 @@ static int cxl_pmem_region_probe(struct device *dev)
                        dev_dbg(dev, "[%d]: %s: no cxl_nvdimm found\n", i,
                                dev_name(&cxlmd->dev));
                        rc = -ENODEV;
-                       goto err;
+                       goto out_nvd;
                }

                /* safe to drop ref now with bridge lock held */
@@ -498,10 +529,17 @@ static int cxl_pmem_region_probe(struct device *dev)
                        dev_dbg(dev, "[%d]: %s: no nvdimm found\n", i,
                                dev_name(&cxlmd->dev));
                        rc = -ENODEV;
-                       goto err;
+                       goto out_nvd;
                }
-               cxl_nvd->region = cxlr_pmem;
-               get_device(&cxlr_pmem->dev);
+
+               /*
+                * Pin the region per nvdimm device as those may be released
+                * out-of-order with respect to the region, and a single nvdimm
+                * maybe associated with multiple regions
+                */
+               rc = cxl_nvdimm_add_region(cxl_nvd, cxlr_pmem);
+               if (rc)
+                       goto out_nvd;
                m->cxl_nvd = cxl_nvd;
                mappings[i] = (struct nd_mapping_desc) {
                        .nvdimm = nvdimm,
```

```
@@ -527,27 +565,18 @@ static int cxl_pmem_region_probe(struct device *dev)
                nvdimm_pmem_region_create(cxl_nvb->nvdimm_bus, &ndr_desc);
        if (!cxlr_pmem->nd_region) {
                rc = -ENOMEM;
-               goto err;
+               goto out_nvd;
        }

        rc = devm_add_action_or_reset(dev, unregister_nvdimm_region,
                                      cxlr_pmem->nd_region);
-out:
+out_nvd:
        kfree(info);
+out_nvb:
        device_unlock(&cxl_nvb->dev);
        put_device(&cxl_nvb->dev);

        return rc;
-
-err:
-       dev_dbg(dev, "failed to create nvdimm region\n");
-       for (i--; i >= 0; i--) {
-               nvdimm = mappings[i].nvdimm;
-               cxl_nvd = nvdimm_provider_data(nvdimm);
-               put_device(&cxl_nvd->region->dev);
-               cxl_nvd->region = NULL;
-       }
-       goto out;
 }

 static struct cxl_driver cxl_pmem_region_driver = {
```