An Introduction to Stacked Git

Myron F. Stowe

Kernel: Platform Enablement Team
Red Hat, Inc.
Westford, Massachusetts
myron.stowe@gmail.com

Technical Presentation/Tutorial

August 18, 2015

Abstract: *Software development is rarely a straightforward, sequential process. More often than not, it's a disordered, iterative undertaking with an end result in mind. Stacked Git (StGit) lends itself well towards this type of environment, helping control a potentially chaotic process by providing a set of commands that help manage both patch series and their individual components. With StGit, one can easily develop and manage patch series that consist of hundreds of individual patches.*

This paper provides an introduction to StGit, demonstrating its use and capabilities by recreating a past kernel development effort that produced a four patch series. This approach provides a concrete example that can be presented as a technical talk and also read individually - optionally working through the demonstration - as a self-paced tutorial.

Keywords: Git, git, Linux, patch, patches, patch development, patch series, software development, Stacked Git, StGit, stg.

This technical presentation/tutorial will be presented during the 2015 *LinuxCon North America* event. All materials necessary to work through the demonstration's example are included in a tar archive that is available at *https://goo.gl/pAKeqe*

Table of Contents

Introduction

"StGit is a Python application[1] providing similar functionality to Quilt (i.e. pushing/popping patches to/from a stack) on top of Git. These operations are performed using Git commands and the patches are stored as Git commit objects, allowing easy merging of the StGit patches into other repositories using standard Git functionality. StGit is licensed under the GNU General Public License."[1]

"Note that StGit is not a source configuration management (SCM) interface on top of Git and it expects a previously initialized Git repository."[1] As such, this tutorial assumes pre-requisite knowledge of Git's basics.

StGit uses the same configuration mechanisms as Git. Reference man pages *git(1)* and *git-config(1)* for more details.

For a full list of StGit commands:

```
$ stg help
```

For quick help on individual subcommands:

```
$ stg help <cmd>
```

For more extensive help on a subcommand:

```
$ man stg-<cmd>
```

Overview

For demonstration purposes we'll utilize an existing patch series that Shuah Khan produced and submitted to the "*linux-pci*" mail list on 02.17.2013.

```
[PATCH v2 0/4] pci: Add PCI_BUS() and PCI_DEVID() interfaces to return bus number and device id
  [PATCH v2 1/4] pci: Add PCI_BUS_NUM() and PCI_DEVID() interfaces to return bus number and …
  [PATCH v2 2/4] pci/aer: Remove local PCI_BUS() define and use PCI_BUS() from pci
  [PATCH v2 3/4] iommu/amd: Remove local PCI_BUS() define and use PCI_BUS() from pci
  [PATCH v2 4/4] iommu/amd: Remove calc_devid() and use PCI_DEVID() from pci
```

The objective of the series was to re-factor multiple, private, instances of `PCI_BUS()`, along with a private instance of `calc_devid()`, into `PCI_BUS_NUM()` and `PCI_DEVID()` interfaces for use globally within the kernel (see Appendix A).

Using this series, we'll re-enact a typical development workflow, including mistakes and moments of insight, acting as if we were developing the series ourselves from scratch.[2]

As mentioned, StGit is not a stand-alone program; it operates on an existing Git repository that has already been created, using `git init` or `git clone`. As the series of interest was developed for the Linux kernel, we'll use a local kernel repository cloned from the kernel.org site.

```
$ cd ~/temp/linux/
$ git remote -v
origin  git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git (fetch)
origin  git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git (push)
```

[1]
```
$ which stg
/usr/bin/stg
$ file /usr/bin/stg
/usr/bin/stg: Python script, ASCII text executable
```

[2] While the example comes from development efforts within the Linux Kernel project, StGit is applicable to any project that utilizes Git - including non-software related projects.

Workflow[3]

Creating a Topic Branch

It's standard practice to create and use "topic" branches when developing patches. By default the new branch will be based off the source branch's current HEAD. Note, however, that for this demonstration we are re-enacting the development of a series that already exists so we'll need to create a branch whose basis is prior to that existing content.

The patch series of interest was pulled upstream from the PCI maintainer's "*pci/shuah-defines*" branch via merge commit 8332606

```
$ git show 8332606
commit 833260631178aa26c70a0b05eabc953f5e47167d
Merge: f6161aa 6f2729b
Author: Bjorn Helgaas <bhelgaas@google.com>
Date:   Fri Mar 29 08:55:26 2013 -0600

    Merge branch 'pci/shuah-defines' into next
…

$ git log --oneline f6161aa..6f2729b
6f2729b iommu/amd: Remove calc_devid() and use PCI_DEVID() from PCI
c5081cd iommu/amd: Remove local PCI_BUS() define and use PCI_BUS_NUM() from PCI
fff0ee3 PCI/AER: Remove local PCI_BUS() define and use PCI_BUS_NUM()from PCI
8546713 PCI: Add PCI_BUS_NUM() and PCI_DEVID() interfaces
```

and the commits occurred during the kernel's v3.10-rc1 merge window.

```
$ git tag --contains 8546713
...
v3.10-rc1     # Earliest tag corresponding to commit 8546713
...
```

As such, we'll create our topic branch using a basis corresponding to the latest released kernel prior to v3.10-rc1; kernel version 'v3.9'.

A branch can be created for use with StGit using either Git or StGit. If the branch is created using Git, we'll also need to subsequently initialize it for use with StGit using `stg init`, initializing its meta-data.

```
$ git checkout -b stg_demo v3.9
Switched to a new branch 'stg_demo'
$ stg init
```

The StGit equivalent of the above two commands is "`stg branch -c stg_demo v3.9`".

We can see what branches currently exist using either Git or StGit:

```
$ git branch      # The StGit equivalent is "stg branch -l"
  master
* stg_demo
```

The asterisk ('*') denotes the current branch.

We now have a branch, whose basis is kernel version 3.9, with which to start reenacting the patch series' development.

```
$ git log -1
commit c1be5a5b1b355d40e6cf79cc979eb66dafa24ad1
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date:   Sun Apr 28 17:36:01 2013 -0700

    Linux 3.9
```

[3] For recommended workflows and best practices see *gitworkflows(7)*.

Creating Patches

With the topic branch in place, we're ready to begin creating patches. Let's create the initial patch, which we'll call "*first.patch*". `stg new [options] [--] [<name>]` will create a new, empty patch, on top of any currently applied patches, making it the new top of the stack:

```
$ stg new first.patch
>> First patch of stg_demo series
```

At this point, the patch is empty, so we need to implement the intended modifications.

The intention of the first part of the series is to convert the private definition and usages of `PCI_BUS()` with a kernel global version called `PCI_BUS_NUM()`. For the first patch, we'll remove the private `PCI_BUS() #define` within *./drivers/pci/pcie/aer/aerdrv_core.c* and also convert the file's `PCI_BUS()` usage to `PCI_BUS_NUM()`.

```
$ vim ./drivers/pci/pcie/aer/aerdrv_core.c
# patch -p1 < ~/temp/stg_demo/commits/fff0ee3.patch[4]
```

While we've implemented the changes corresponding to the patch we want, at this point the commit object, which can be displayed by `stg show`, only includes the "commit message" we supplied with `stg new`.

```
$ stg show            # Note working directory vs. index changes
commit 57776b7eb636a7802d9b9cb356197015bf52e73f
Author: Myron Stowe <myron.stowe@redhat.com>
Date:   Fri Jul 31 10:52:31 2015 -0600

    First patch of stg_demo series
```

`stg show` only displays the commit object corresponding to the patch. It does not show changes in either the working directory or in Git's index. The following commands help clarify the distinction:

```
$ git status          # Indicates file is modified (in the working directory) but not staged
On branch stg_demo
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   drivers/pci/pcie/aer/aerdrv_core.c

no changes added to commit (use "git add" and/or "git commit -a")
$ git diff
...                   # Changes shown in unified diff format (see Appendix A A.2)
$ git diff --staged   # No changes shown (Git's index has not been changed/updated)
$ stg status                                  # "stg status" is an alias for "git status -s"
 M drivers/pci/pcie/aer/aerdrv_core.c         # XY - X=index, Y=work tree[5]
```

With the changes in place we generate a new commit object for the patch using `stg refresh [options] [--] [<files or dirs>]`:

```
$ stg refresh
Now at patch "first.patch"
```

Now the patch's changes can be seen using `stg show`, and since the patch is also a regular Git commit, we can see it with Git's regular tools using, say, `git log`.

```
$ stg show
...           # Commit object (see Appendix A A.2)
$ git log -p -1
...           # Commit object (see Appendix A A.2)
```

[4] During development we obviously use an editor to effect the intended changes to the file(s). For this demonstration's purposes, the patch files have been included in the tutorial's tar archive. With such, the result of editing by enacting this command can be emulated.

[5] See *git-status(1)*; specifically the "Short Format" section.

Having just created this patch, we now realize that we should have introduced the new `#define` of `PCI_BUS_NUM()` as a separate, precursor patch. Oversights such as this are common occurrences during development of new material, and this is where the real power and benefit of StGit comes to light. Let's effectively backup and create what should have been the first patch of the series.

First, we may want to rename the existing patch before creating another using `stg rename [options] [--] [oldpatch] <newpatch>`:

```
$ stg rename first.patch second.patch
Renaming patch "first.patch" to "second.patch" ... done
```

We may also want to modify the current working commit message of the existing patch using `stg edit [options] [--] [<patch>]`:

```
$ stg edit    # To see the unified diff formatted changes also, add the '-d' option
>> "First" -> "Second"
$ stg show
...           # Commit object with "First" having been changed to "Second"
```

Continuing with creating what should have originally been the first patch of the series, create a new patch with `stg new`, make the intended modifications constituting the patch, and generate a new commit object from the patch with the following:

```
$ stg new -m "First patch of stg_demo series" first.patch
Now at patch "first.patch"
$ vim ./include/linux/pci.h
# patch -p1 < ~/temp/stg_demo/commits/8546713.patch[4]
$ stg refresh
Now at patch "first.patch"
```

The three command sequences "`stg new [<name>]`", "`vim <file(s)>`", "`stg refresh`", and possibly numerous iterations thereof, represent a typical cycle of StGit based patch development.

Manipulating Patch Ordering on the Stack

At this point, we currently have a two patch series consisting of *first.patch* and *second.patch* which we can see using `stg series`:

```
$ stg series
+ second.patch
> first.patch
```

StGit shows patches chronologically ordered from top to bottom. Note that this is the opposite from `git log`, which by default uses reverse chronological ordering. So, while we've succeeded in creating our first two patches, they are currently out of order. As StGit patches are stacked based, we can easily correct the ordering by manipulating the placement of the patches on the stack.

The most basic StGit stack commands are `pop` and `push` which can be used to manipulate the ordering of patches within the stack. To correct the ordering of our two patches, we can do the following:

```
$ stg pop -a                  # stg pop [-n N]
Popped first.patch -- second.patch
No patch applied
$ stg push first.patch
Pushing patch "first.patch" ... done
Now at patch "first.patch"
$ stg push                    # stg push [<patch_name>] -or- stg push [-n N] -or- stg push -a
Pushing patch "second.patch" ... done
Now at patch "second.patch"
```

Now the patches are in correct order with the latest, top-most patch being *second.patch* as can be seen using `stg top`.

```
$ stg top
second.patch
```

Let's continue our series' development after a break or interruption has occurred, so: "Where were we?" Reviewing what's currently on the stack we see:

```
$ stg series --description
+ first.patch  # First patch of stg_demo series
> second.patch # Second patch of stg_demo series
```

Now create the last patch of the series. As before, create a new patch with `stg new`, make the intended modifications that make up the patch, followed by generating a new commit object from the patch.

```
$ stg new -m "Third patch of stg_demo series" third.patch
Now at patch "third.patch"
$ vim ./drivers/iommu/amd_iommu{,_init}.c ./drivers/iommu/amd_iommu_types.h
# patch -p1 < ~/temp/stg_demo/commits/combined.patch[4]
$ stg refresh
Now at patch "third.patch"
```

This patch was more involved: it modifies more than just a single file. We can show the files modified by a patch, optionally along with their `diffstat` histogram data, using `stg files [options] [--] [[<branch>:]<patch>]`.

```
$ stg files -s
drivers/iommu/amd_iommu.c       |   14 +++++++-------
drivers/iommu/amd_iommu_init.c  |   40 ++++++++++++++++++++----------------
drivers/iommu/amd_iommu_types.h |   11 +++++-----
3 files changed, 28 insertions(+), 37 deletions(-)
```

After some thought, we realize that it would be better if this last patch were re-factored a bit, splitting it up into two distinct patches to isolate the two independent types of changes that are occurring.

Exporting and Importing Patches

As we currently have a valid, working patch set, let's keep this series intact and try out the re-factoring idea with a separate 'v2' branch just in case it doesn't work out.

We need to save our current work, create a new 'v2' topic branch for our re-factoring idea, and somehow get our patches onto the new branch. We've already seen how to create a new topic branch, but we need an efficient way to save or current work and populate our new branch with the current set of patches we've developed. StGit has commands to both export and import patches. `stg export [options][--] [<patch1>] [<patch2>] [<patch3>..<patch4>]` exports a range of applied patches to a given directory. Conversely, `stg import [options] [--] [<file>|<url>]`, specifically using the '`-s`' option, will import a series of patches from a *series* file or a tar archive. Putting everything together we get:

```
$ mkdir /tmp/stg_demo_v1/                  # Create a dir to archive our working set of patches
$ stg export -d /tmp/stg_demo_v1/ first.patch..third.patch
Checking for changes in the working directory ... done
$ git checkout -b stg_demo_v2 v3.9
Switched to a new branch 'stg_demo_v2'
$ stg init
$ stg import -s /tmp/stg_demo_v1/series    # Populate our new branch with the working series
Checking for changes in the working directory ... done
Importing patch "first.patch" ... done
Importing patch "second.patch" ... done
Importing patch "third.patch" ... done
Now at patch "third.patch"
```

We have exported (saved off) our current work, created a 'v2' branch, initialized it for use with StGit, and imported our existing patches into it. We are effectively right where we were before, but on a different branch, so let's continue with our re-factoring thought that splits up the third patch.

Splitting a Patch into Multiple Patches

There are a number of ways in which we can split out the two independent types of changes currently combined within the third patch. The following workflow is what I've come to use over time for creating multiple commits from a single set of existing edits or patch:

```
$ stg rename third.patch fourth.patch
Renaming patch "third.patch" to "fourth.patch" ... done
$ stg pop
Popped fourth.patch
Now at patch "second.patch"
$ patch -p1 < /tmp/stg_demo_v1/third.patch
patching file drivers/iommu/amd_iommu.c
patching file drivers/iommu/amd_iommu_init.c
patching file drivers/iommu/amd_iommu_types.h
    $ stg new third.patch
    Invoking the editor: "vim .stgit-new.txt" ... done
    Now at patch "third.patch"
    $ git add --patch ./drivers/iommu/amd_iommu.c
    >> n, y, y                       # Accept hunks related to PCI_BUS->PCI_BUS_NUM
    $ git add --patch ./drivers/iommu/amd_iommu_init.c
    >> n, y (12x), n, y              # Accept hunks related to PCI_BUS->PCI_BUS_NUM
    $ git add --patch ./drivers/iommu/amd_iommu_types.h
    >> y, y, n                       # Accept hunks related to PCI_BUS->PCI_BUS_NUM
    $ git stash --keep-index         # Discard, but keep (stash), working dir changes
    Saved working directory and index state WIP on stg_demo_v2: 598ba8c Third
    patch of stg_demo series
    HEAD is now at 598ba8c Third patch of stg_demo series
    # build/test
    $ stg refresh
    Now at patch "third.patch"
    [$ git stash pop] # Only needed if more intermediate split-out patches are intended
$ stg push
Pushing patch "fourth.patch" ... done (modified)
Now at patch "fourth.patch"
```

To set up the patch separation sequencing, we rename the patch we're going to split up to a name representing the last patch of the separation effort. Next we pop it off of the stack temporarily so that we can create the new intermediate disconnected patch(es) in the correct place within the stack. We then copy the content of the original coalesced patch into our working directory. Note that, at this point, the sets of changes are only in our working directory, they are not staged in Git's index.

We are now prepared to begin sequencing through the N-1 intermediate patches of the separation effort (in our case we are splitting a single patch into two (N = 2) so there is only one intermediate patch – *third.patch*. While this is the simplest case, the algorithm works for any value of N). Iterate through the N-1 intermediate patches as follows:[6]

> Create a new intermediate patch using `stg new` and stage only the subset of changes wanted using the interactive form of `git add` (`git add -p`). After this step our working directory will have all the patch hunks that remain to be separated and Git's staging area will have the selected subset of hunks that are going to constitute the current, intermediate patch.
>
> Run `git stash --keep-index` to save and undo the outstanding, unstaged (working directory) changes while leaving the staged changes in Git's index, resetting the working tree to match the index.
>
> Now generate a new commit for the current intermediate patch with `stg refresh`.
>
> If there are more intermediate splits remaining, restore the remaining changes with `git stash pop` and repeat the iteration sequence as many times as needed.

When the final patch in the split-out effort is reached, just run `stg push` and let Git do all of the work of recognizing and removing the patch hunks that were incorporated in the split-out effort's prior patches - leaving us with only the remaining hunks intended as content for the split-out effort's final patch.

[6] The sequence to iterate upon is denoted by indentation – both in the code and the corresponding text description.

Cleaning Up the Series for Public Submission

We are effectively done with the series development. This is a good time to create public commit messages for each of the patches of the series. Good commit messages are critical: they should describe exactly what the patch accomplishes in a manner that is succinct and understandable by others. If the description is too complicated and referencing too many discrete ideas then that's a strong indication that the patch should be split.

Edit the descriptions replacing the personal working notes with pertinent, meaningful descriptions and anything else that the project requires, such as adding "`Signed-off-by:`", to ready the patches for submission.

```
$ stg edit first.patch
>> Commit's title, message (description), and anything else required by the project
$ stg edit second.patch
>> Commit's title, message (description), and anything else required by the project
$ stg edit third.patch
>> Commit's title, message (description), and anything else required by the project
$ stg edit        # No need to specify <patch> when targeting current (top of stack) patch
>> Commit's title, message (description), and anything else required by the project
```

Rebasing a Patch Series

Another aspect to consider at this point of development is rebasing. We may have been working on this series for a long time, and in that time our branch's basis may have new content. We should bring in any new content from our originating branch that has occurred in the interim (rebase) to make sure that no new conflicts exist.[7]

To refresh a topic branch:

```
$ git remote update origin
$ stg rebase origin/master
# Resolve any conflicts if they occur
```

Submitting Patches Upstream

Finally, take care of any procedures that should be done or are required by the project prior to submitting. For the kernel, this includes checking that the patches meet the kernel's guidelines by passing them through *./scripts/checkpatch.pl*.

We are now ready to submit our patch series to the appropriate mail list(s), maintainer, or mainline. It's always a good idea to initially mail them to yourself first as part of a final check that everything is in place and ready to publish.

```
$ stg mail --to=<email_addr> [--cc=<email_addr>] [--prefix="RFC"] [--version="v2"]  \
[-c ~/temp/stg_demo/commits/series.cover] <first_stg_patch..last_stg_patch>
```

Removing the Topic Branch

At some point we'll probably want to remove our topic branch. To remove our topic branch we first cleanup the StGit meta-data using `stg branch –cleanup [--force] [--] [<branch>]` (basically the counterpart to `stg init`). We then complete the removal using `git branch (-d | -D) <branchname>`.

If our topic branch has not been "fully merged" into another branch we'll need to force the removal by specifying the '`-D`' option to `git branch`.

```
$ git checkout master
$ stg branch --cleanup --force stg_demo
$ git branch -D stg_demo
$ stg branch --cleanup --force stg_demo_v2
$ git branch -D stg_demo_v2
```

[7] Note that rebasing doesn't apply to our specific situation as our branch was derived from a past point, and not the HEAD, of the originating branch.

Reference Material

The following references may be helpful while becoming acquainted with StGit:

gitworkflows(7)

https://gna.org/projects/stgit

http://www.procode.org/stgit/

http://www.procode.org/stgit/doc/stg.html

http://www.procode.org/stgit/doc/tutorial.html

./<stg_source_tree_basedir>/Documentation/tutorial.txt[8]

Bibliography

[1] ["Stacked Git – Summary" page], (n.d.), Retrieved from *https://gna.org/projects/stgit* (also found at *www.procode.org/stgit/*).

[8] StGit source is available at *git://repo.or.cz/w/stgit.git*

Appendix A Example's Patch Series

The patch series used as example material in this paper comes from the Linux Kernel project. The series' patches provide concrete components that anyone wanting to learn StGit can easily acquire and use. While the series may not seem so at first due to its combined size, its patches were chosen because they are conceptually very simple – the intention of the paper is to provide an introduction to StGit, not focus on the patches themselves. Conceptually, they accomplish two tasks:

- Create a kernel global version of `PCI_BUS_NUM()`,
  remove the locally private versions of `PCI_BUS()`, and convert its usages to `PCI_BUS_NUM()`.
- Create a kernel global version of `PCI_DEVID()`,
  remove the local private version of `calc_devid()`, and convert its usages to `PCI_DEVID()`.

```
[PATCH v2 0/4] pci: Add PCI_BUS() and PCI_DEVID() interfaces to return bus number and device id

pci defines PCI_DEVFN(), PCI_SLOT(), and PCI_FUNC() interfaces, however,
it doesn't have interfaces to return PCI bus and PCI device id. Drivers
(AMD IOMMU, and AER) have module specific definitions for PCI_BUS() and
AMD_IOMMU driver also has a module specific interface to calculate PCI
device id from bus number and devfn.

This patch set adds PCI_BUS_NUM(), and PCI_DEVID() to pci.h, changes AER
to use PCI_BUS_NUM() from pci and remove local PCI_BUS() define. Changes
AMD_IOMMU driver to use PCI_BUS_NUM() and PCI_DEVID() from pci and remove
local PCI_BUS() define and local PCI_DEVID() implementation.


[PATCH v2 1/4] pci: Add PCI_BUS_NUM() and PCI_DEVID() interfaces to return bus number and device id
  commit 85467136cdcc674f30beb0e5b79f048fe3a6a76f
  PCI: Add PCI_BUS_NUM() and PCI_DEVID() interfaces
  include/linux/pci.h | 15 +++++++++++++++
  1 file changed, 15 insertions(+)

[PATCH v2 2/4] pci/aer: Remove local PCI_BUS() define and use PCI_BUS() from pci
  commit fff0ee3640d55c1df4e9da6084b20e2b3345abec
  PCI/AER: Remove local PCI_BUS() define and use PCI_BUS_NUM() from PCI
  drivers/pci/pcie/aer/aerdrv_core.c | 4 +---
  1 file changed, 1 insertion(+), 3 deletions(-)

[PATCH v2 3/4] iommu/amd: Remove local PCI_BUS() define and use PCI_BUS() from pci
  commit c5081cd7a2b3db31a72b4c697321cf0425dbc2f1
  iommu/amd: Remove local PCI_BUS() define and use PCI_BUS_NUM() from PCI
  drivers/iommu/amd_iommu.c       | 12 ++++++------
  drivers/iommu/amd_iommu_init.c  | 34 +++++++++++++++++-----------------
  drivers/iommu/amd_iommu_types.h |  4 +---
  3 files changed, 24 insertions(+), 26 deletions(-)

[PATCH v2 4/4] iommu/amd: Remove calc_devid() and use PCI_DEVID() from pci
  commit 6f2729bab2cc386bb603698646dacd9ab6297ba0
  iommu/amd: Remove calc_devid() and use PCI_DEVID() from PCI
  drivers/iommu/amd_iommu.c       | 2 +-
  drivers/iommu/amd_iommu_init.c  | 6 +++---
  drivers/iommu/amd_iommu_types.h | 7 -------
  3 files changed, 4 insertions(+), 11 deletions(-)
```

A.1 commit 8546713

```
commit 85467136cdcc674f30beb0e5b79f048fe3a6a76f
Author: Shuah Khan <shuah.khan@hp.com>
Date:   Wed Feb 27 17:06:45 2013 -0700

    PCI: Add PCI_BUS_NUM() and PCI_DEVID() interfaces

    PCI defines PCI_DEVFN(), PCI_SLOT(), and PCI_FUNC() interfaces; however,
    it doesn't have interfaces to return PCI bus and PCI device id. Drivers
    (AMD IOMMU, and AER) implement module specific definitions for PCI_BUS()
    and AMD_IOMMU driver also has a module specific interface to calculate PCI
    device id from bus number and devfn.

    Add PCI_BUS_NUM and PCI_DEVID interfaces to return PCI bus number and PCI
    device id respectively to avoid the need for duplicate definitions in other
    modules. AER driver code and AMD IOMMU driver define PCI_BUS. AMD IOMMU
    driver defines an interface to calculate device id from bus number, and
    devfn pair.

    PCI_DEVFN(), PCI_SLOT(), and PCI_FUNC() interfaces are exported to
    user-space via uapi/linux/pci.h. However, in the interest to keep the new
    interfaces as kernel only and not export them to user-space unnecessarily,
    added them to linux/pci.h instead.

    Signed-off-by: Shuah Khan <shuah.khan@hp.com>
    Signed-off-by: Bjorn Helgaas <bhelgaas@google.com>
    Acked-by: Joerg Roedel <joro@8bytes.org>

diff --git a/include/linux/pci.h b/include/linux/pci.h
index 2461033a..849a336 100644
--- a/include/linux/pci.h
+++ b/include/linux/pci.h
@@ -35,6 +35,21 @@
 /* Include the ID list */
 #include <linux/pci_ids.h>

+/*
+ * The PCI interface treats multi-function devices as independent
+ * devices.  The slot/function address of each device is encoded
+ * in a single byte as follows:
+ *
+ *     7:3 = slot
+ *     2:0 = function
+ * PCI_DEVFN(), PCI_SLOT(), and PCI_FUNC() are defined uapi/linux/pci.h
+ * In the interest of not exposing interfaces to user-space unnecessarily,
+ * the following kernel only defines are being added here.
+ */
+#define PCI_DEVID(bus, devfn)  ((((u16)bus) << 8) | devfn)
+/* return bus from PCI devid = ((u16)bus_number) << 8) | devfn */
+#define PCI_BUS_NUM(x) (((x) >> 8) & 0xff)
+
 /* pci_slot represents a physical slot */
 struct pci_slot {
        struct pci_bus *bus;            /* The bus this slot is on */
```

A.2 commit fff0ee3

```
commit fff0ee3640d55c1df4e9da6084b20e2b3345abec
Author: Shuah Khan <shuah.khan@hp.com>
Date:   Wed Feb 27 17:07:05 2013 -0700

    PCI/AER: Remove local PCI_BUS() define and use PCI_BUS_NUM() from PCI

    Change to remove local PCI_BUS() define and use the new PCI_BUS_NUM()
    interface from PCI.

    Signed-off-by: Shuah Khan <shuah.khan@hp.com>
    Signed-off-by: Bjorn Helgaas <bhelgaas@google.com>
    Acked-by: Joerg Roedel <joro@8bytes.org>

diff --git a/drivers/pci/pcie/aer/aerdrv_core.c b/drivers/pci/pcie/aer/aerdrv_core.c
index 564d97f..8ec8b4f 100644
--- a/drivers/pci/pcie/aer/aerdrv_core.c
+++ b/drivers/pci/pcie/aer/aerdrv_core.c
@@ -89,8 +89,6 @@ static int add_error_device(struct aer_err_info *e_info, struct pci_dev *dev)
        return -ENOSPC;
 }

-#define        PCI_BUS(x)      (((x) >> 8) & 0xff)
-
 /**
  * is_error_source - check whether the device is source of reported error
  * @dev: pointer to pci_dev to be checked
@@ -106,7 +104,7 @@ static bool is_error_source(struct pci_dev *dev, struct aer_err_info *e_info)
         * When bus id is equal to 0, it might be a bad id
         * reported by root port.
         */
-       if (!nosourceid && (PCI_BUS(e_info->id) != 0)) {
+       if (!nosourceid && (PCI_BUS_NUM(e_info->id) != 0)) {
                /* Device ID match? */
                if (e_info->id == ((dev->bus->number << 8) | dev->devfn))
                        return true;
```

A.3 commit c5081cd

```
commit c5081cd7a2b3db31a72b4c697321cf0425dbc2f1
Author: Shuah Khan <shuah.khan@hp.com>
Date:   Wed Feb 27 17:07:19 2013 -0700

    iommu/amd: Remove local PCI_BUS() define and use PCI_BUS_NUM() from PCI

    Change to remove local PCI_BUS() define and use the new PCI_BUS_NUM()
    interface from PCI.

    Signed-off-by: Shuah Khan <shuah.khan@hp.com>
    Signed-off-by: Bjorn Helgaas <bhelgaas@google.com>
    Acked-by: Joerg Roedel <joro@8bytes.org>

diff --git a/drivers/iommu/amd_iommu.c b/drivers/iommu/amd_iommu.c
index 98f555d..2a50fbe 100644
--- a/drivers/iommu/amd_iommu.c
+++ b/drivers/iommu/amd_iommu.c
@@ -649,26 +649,26 @@ retry:
        case EVENT_TYPE_ILL_DEV:
                printk("ILLEGAL_DEV_TABLE_ENTRY device=%02x:%02x.%x "
                       "address=0x%016llx flags=0x%04x]\n",
-                      PCI_BUS(devid), PCI_SLOT(devid), PCI_FUNC(devid),
+                      PCI_BUS_NUM(devid), PCI_SLOT(devid), PCI_FUNC(devid),
                       address, flags);
                dump_dte_entry(devid);
                break;
        case EVENT_TYPE_IO_FAULT:
                printk("IO_PAGE_FAULT device=%02x:%02x.%x "
                       "domain=0x%04x address=0x%016llx flags=0x%04x]\n",
-                      PCI_BUS(devid), PCI_SLOT(devid), PCI_FUNC(devid),
+                      PCI_BUS_NUM(devid), PCI_SLOT(devid), PCI_FUNC(devid),
                       domid, address, flags);
                break;
        case EVENT_TYPE_DEV_TAB_ERR:
                printk("DEV_TAB_HARDWARE_ERROR device=%02x:%02x.%x "
                       "address=0x%016llx flags=0x%04x]\n",
-                      PCI_BUS(devid), PCI_SLOT(devid), PCI_FUNC(devid),
+                      PCI_BUS_NUM(devid), PCI_SLOT(devid), PCI_FUNC(devid),
                       address, flags);
                break;
        case EVENT_TYPE_PAGE_TAB_ERR:
                printk("PAGE_TAB_HARDWARE_ERROR device=%02x:%02x.%x "
                       "domain=0x%04x address=0x%016llx flags=0x%04x]\n",
-                      PCI_BUS(devid), PCI_SLOT(devid), PCI_FUNC(devid),
+                      PCI_BUS_NUM(devid), PCI_SLOT(devid), PCI_FUNC(devid),
                       domid, address, flags);
                break;
        case EVENT_TYPE_ILL_CMD:
@@ -682,13 +682,13 @@ retry:
        case EVENT_TYPE_IOTLB_INV_TO:
                printk("IOTLB_INV_TIMEOUT device=%02x:%02x.%x "
                       "address=0x%016llx]\n",
-                      PCI_BUS(devid), PCI_SLOT(devid), PCI_FUNC(devid),
+                      PCI_BUS_NUM(devid), PCI_SLOT(devid), PCI_FUNC(devid),
                       address);
                break;
        case EVENT_TYPE_INV_DEV_REQ:
                printk("INVALID_DEVICE_REQUEST device=%02x:%02x.%x "
                       "address=0x%016llx flags=0x%04x]\n",
-                      PCI_BUS(devid), PCI_SLOT(devid), PCI_FUNC(devid),
+                      PCI_BUS_NUM(devid), PCI_SLOT(devid), PCI_FUNC(devid),
                       address, flags);
                break;
        default:
diff --git a/drivers/iommu/amd_iommu_init.c b/drivers/iommu/amd_iommu_init.c
index b6ecddb..f8ed6f1 100644
--- a/drivers/iommu/amd_iommu_init.c
```

```
+++ b/drivers/iommu/amd_iommu_init.c
@@ -423,7 +423,7 @@ static int __init find_last_devid_from_ivhd(struct ivhd_header *h)
        p += sizeof(*h);
        end += h->length;

-       find_last_devid_on_pci(PCI_BUS(h->devid),
+       find_last_devid_on_pci(PCI_BUS_NUM(h->devid),
                        PCI_SLOT(h->devid),
                        PCI_FUNC(h->devid),
                        h->cap_ptr);
@@ -784,10 +784,10 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,

                        DUMP_printk("  DEV_ALL\t\t\t first devid: %02x:%02x.%x"
                                    " last device %02x:%02x.%x flags: %02x\n",
-                                   PCI_BUS(iommu->first_device),
+                                   PCI_BUS_NUM(iommu->first_device),
                                    PCI_SLOT(iommu->first_device),
                                    PCI_FUNC(iommu->first_device),
-                                   PCI_BUS(iommu->last_device),
+                                   PCI_BUS_NUM(iommu->last_device),
                                    PCI_SLOT(iommu->last_device),
                                    PCI_FUNC(iommu->last_device),
                                    e->flags);
@@ -801,7 +801,7 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,

                        DUMP_printk("  DEV_SELECT\t\t\t devid: %02x:%02x.%x "
                                    "flags: %02x\n",
-                                   PCI_BUS(e->devid),
+                                   PCI_BUS_NUM(e->devid),
                                    PCI_SLOT(e->devid),
                                    PCI_FUNC(e->devid),
                                    e->flags);
@@ -813,7 +813,7 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,

                        DUMP_printk("  DEV_SELECT_RANGE_START\t "
                                    "devid: %02x:%02x.%x flags: %02x\n",
-                                   PCI_BUS(e->devid),
+                                   PCI_BUS_NUM(e->devid),
                                    PCI_SLOT(e->devid),
                                    PCI_FUNC(e->devid),
                                    e->flags);
@@ -827,11 +827,11 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,

                        DUMP_printk("  DEV_ALIAS\t\t\t devid: %02x:%02x.%x "
                                    "flags: %02x devid_to: %02x:%02x.%x\n",
-                                   PCI_BUS(e->devid),
+                                   PCI_BUS_NUM(e->devid),
                                    PCI_SLOT(e->devid),
                                    PCI_FUNC(e->devid),
                                    e->flags,
-                                   PCI_BUS(e->ext >> 8),
+                                   PCI_BUS_NUM(e->ext >> 8),
                                    PCI_SLOT(e->ext >> 8),
                                    PCI_FUNC(e->ext >> 8));

@@ -846,11 +846,11 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,
                        DUMP_printk("  DEV_ALIAS_RANGE\t\t "
                                    "devid: %02x:%02x.%x flags: %02x "
                                    "devid_to: %02x:%02x.%x\n",
-                                   PCI_BUS(e->devid),
+                                   PCI_BUS_NUM(e->devid),
                                    PCI_SLOT(e->devid),
                                    PCI_FUNC(e->devid),
                                    e->flags,
-                                   PCI_BUS(e->ext >> 8),
+                                   PCI_BUS_NUM(e->ext >> 8),
                                    PCI_SLOT(e->ext >> 8),
                                    PCI_FUNC(e->ext >> 8));

@@ -864,7 +864,7 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,
```

```
                        DUMP_printk("  DEV_EXT_SELECT\t\t devid: %02x:%02x.%x "
                                    "flags: %02x ext: %08x\n",
-                                   PCI_BUS(e->devid),
+                                   PCI_BUS_NUM(e->devid),
                                    PCI_SLOT(e->devid),
                                    PCI_FUNC(e->devid),
                                    e->flags, e->ext);
@@ -877,7 +877,7 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,

                        DUMP_printk("  DEV_EXT_SELECT_RANGE\t devid: "
                                    "%02x:%02x.%x flags: %02x ext: %08x\n",
-                                   PCI_BUS(e->devid),
+                                   PCI_BUS_NUM(e->devid),
                                    PCI_SLOT(e->devid),
                                    PCI_FUNC(e->devid),
                                    e->flags, e->ext);
@@ -890,7 +890,7 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,
                case IVHD_DEV_RANGE_END:

                        DUMP_printk("  DEV_RANGE_END\t\t devid: %02x:%02x.%x\n",
-                                   PCI_BUS(e->devid),
+                                   PCI_BUS_NUM(e->devid),
                                    PCI_SLOT(e->devid),
                                    PCI_FUNC(e->devid));

@@ -924,7 +924,7 @@ static int __init init_iommu_from_acpi(struct amd_iommu *iommu,

                        DUMP_printk("  DEV_SPECIAL(%s[%d])\t\tdevid: %02x:%02x.%x\n",
                                    var, (int)handle,
-                                   PCI_BUS(devid),
+                                   PCI_BUS_NUM(devid),
                                    PCI_SLOT(devid),
                                    PCI_FUNC(devid));

@@ -1086,7 +1086,7 @@ static int __init init_iommu_all(struct acpi_table_header *table)

                        DUMP_printk("device: %02x:%02x.%01x cap: %04x "
                                    "seg: %d flags: %01x info %04x\n",
-                                   PCI_BUS(h->devid), PCI_SLOT(h->devid),
+                                   PCI_BUS_NUM(h->devid), PCI_SLOT(h->devid),
                                    PCI_FUNC(h->devid), h->cap_ptr,
                                    h->pci_seg, h->flags, h->info);
                        DUMP_printk("       mmio-addr: %016llx\n",
@@ -1116,7 +1116,7 @@ static int iommu_init_pci(struct amd_iommu *iommu)
        int cap_ptr = iommu->cap_ptr;
        u32 range, misc, low, high;

-       iommu->dev = pci_get_bus_and_slot(PCI_BUS(iommu->devid),
+       iommu->dev = pci_get_bus_and_slot(PCI_BUS_NUM(iommu->devid),
                                          iommu->devid & 0xff);
        if (!iommu->dev)
                return -ENODEV;
@@ -1388,8 +1388,8 @@ static int __init init_unity_map_range(struct ivmd_header *m)

        DUMP_printk("%s devid_start: %02x:%02x.%x devid_end: %02x:%02x.%x"
                    " range_start: %016llx range_end: %016llx flags: %x\n", s,
-                   PCI_BUS(e->devid_start), PCI_SLOT(e->devid_start),
-                   PCI_FUNC(e->devid_start), PCI_BUS(e->devid_end),
+                   PCI_BUS_NUM(e->devid_start), PCI_SLOT(e->devid_start),
+                   PCI_FUNC(e->devid_start), PCI_BUS_NUM(e->devid_end),
                    PCI_SLOT(e->devid_end), PCI_FUNC(e->devid_end),
                    e->address_start, e->address_end, m->flags);

diff --git a/drivers/iommu/amd_iommu_types.h b/drivers/iommu/amd_iommu_types.h
index e38ab43..a07882f 100644
--- a/drivers/iommu/amd_iommu_types.h
+++ b/drivers/iommu/amd_iommu_types.h
@@ -24,6 +24,7 @@
 #include <linux/mutex.h>
 #include <linux/list.h>
 #include <linux/spinlock.h>
```

```
+#include <linux/pci.h>

 /*
  * Maximum number of IOMMUs supported
@@ -315,9 +316,6 @@

 #define MAX_DOMAIN_ID 65536

-/* FIXME: move this macro to <linux/pci.h> */
-#define PCI_BUS(x) (((x) >> 8) & 0xff)
-
 /* Protection domain flags */
 #define PD_DMA_OPS_MASK                (1UL << 0) /* domain used for dma_ops */
 #define PD_DEFAULT_MASK                (1UL << 1) /* domain is a default dma_ops
```

A.4 commit 6f2729b

```
commit 6f2729bab2cc386bb603698646dacd9ab6297ba0
Author: Shuah Khan <shuah.khan@hp.com>
Date:   Wed Feb 27 17:07:30 2013 -0700

    iommu/amd: Remove calc_devid() and use PCI_DEVID() from PCI

    Change to remove calc_devid() and use PCI_DEVID() from PCI instead.

    Signed-off-by: Shuah Khan <shuah.khan@hp.com>
    Signed-off-by: Bjorn Helgaas <bhelgaas@google.com>
    Acked-by: Joerg Roedel <joro@8bytes.org>

diff --git a/drivers/iommu/amd_iommu.c b/drivers/iommu/amd_iommu.c
index 2a50fbe..e046d7a 100644
--- a/drivers/iommu/amd_iommu.c
+++ b/drivers/iommu/amd_iommu.c
@@ -173,7 +173,7 @@ static inline u16 get_device_id(struct device *dev)
 {
        struct pci_dev *pdev = to_pci_dev(dev);

-       return calc_devid(pdev->bus->number, pdev->devfn);
+       return PCI_DEVID(pdev->bus->number, pdev->devfn);
 }

 static struct iommu_dev_data *get_dev_data(struct device *dev)
diff --git a/drivers/iommu/amd_iommu_init.c b/drivers/iommu/amd_iommu_init.c
index f8ed6f1..551768a 100644
--- a/drivers/iommu/amd_iommu_init.c
+++ b/drivers/iommu/amd_iommu_init.c
@@ -406,7 +406,7 @@ static int __init find_last_devid_on_pci(int bus, int dev, int fn, int cap_ptr)
        u32 cap;

        cap = read_pci_config(bus, dev, fn, cap_ptr+MMIO_RANGE_OFFSET);
-       update_last_devid(calc_devid(MMIO_GET_BUS(cap), MMIO_GET_LD(cap)));
+       update_last_devid(PCI_DEVID(MMIO_GET_BUS(cap), MMIO_GET_LD(cap)));

        return 0;
 }
@@ -1128,9 +1128,9 @@ static int iommu_init_pci(struct amd_iommu *iommu)
        pci_read_config_dword(iommu->dev, cap_ptr + MMIO_MISC_OFFSET,
                              &misc);

-       iommu->first_device = calc_devid(MMIO_GET_BUS(range),
+       iommu->first_device = PCI_DEVID(MMIO_GET_BUS(range),
                                         MMIO_GET_FD(range));
-       iommu->last_device = calc_devid(MMIO_GET_BUS(range),
+       iommu->last_device = PCI_DEVID(MMIO_GET_BUS(range),
                                        MMIO_GET_LD(range));

        if (!(iommu->cap & (1 << IOMMU_CAP_IOTLB)))
diff --git a/drivers/iommu/amd_iommu_types.h b/drivers/iommu/amd_iommu_types.h
index a07882f..ec36cf6 100644
--- a/drivers/iommu/amd_iommu_types.h
+++ b/drivers/iommu/amd_iommu_types.h
@@ -701,13 +701,6 @@ extern int amd_iommu_max_glx_val;
   */
 extern void iommu_flush_all_caches(struct amd_iommu *iommu);

-/* takes bus and device/function and returns the device id
- * FIXME: should that be in generic PCI code? */
-static inline u16 calc_devid(u8 bus, u8 devfn)
-{
-       return (((u16)bus) << 8) | devfn;
-}
-
 static inline int get_ioapic_devid(int id)
 {
```

```
struct devid_map *entry;
```