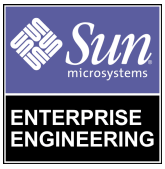
DR Requirements for I/O Device Drivers

By Enrique Vargas - Enterprise Engineering

Sun BluePrints™ Online - June 1999



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-4399-10
Revision 01, June 1999

DR Requirements for I/O Device Drivers

This is the second article in the *Dynamic Reconfiguration* (DR) series. It covers the device driver functions that are required to fully support the DR framework. The examples in this article were extracted from a Starfire™ environment but the device driver requirements apply equally well to the DR technology used on the Sun Enterprise™ 3000 through 6500 servers.

If you have already implemented DR in your Datacenter and are interested in sharing your experiences, or if you would like to see us address a topic that will empower you to take DR to the next level, please give us your feedback through the *"Tell Us"* button at the end of this article.

DR-compliant I/O Device Driver Requirements

To fully support DR operations, all I/O device drivers associated with the system board to be attached or detached must be able to build (attach) and tear down (detach) all device instances spanning from their root nodes. These I/O drivers must fully implement the following Solaris™ DDI/DKI (Device Driver Interface/Device Kernel Interface) entry points as specified in the *Writing Device Drivers* section of the *Driver Developer Site 1.0 AnswerBook™ (*`http://docs.sun.com:80/ab2/@DSCBrowse?driver=1`):

- DDI_ATTACH (`attach(9E)`)
- DDI_DETACH (`detach(9E)`)
- DDI_SUSPEND
- DDI_RESUME

Under the Solaris 2.5.1 and Solaris 2.6 operating environments, Starfire DR requires newly created DR-compliant drivers to register with the `dr` driver by adding the following line to the `/etc/system` file:

```
set dr:detach_safe_list1="driver1 driver2... drivern"
```

In the Solaris 7 operating environment, Starfire DR requires that newly-created DR-compliant drivers set the `D_HOTPLUG` bit in the `cb_ops` data structure.

Device Driver Memory Allocation

An additional requirement to support DR is that drivers which allocate memory directly through the `page_create()`, `page_create_va()`, or other kernel routines, *must not* hold the locked memory pages for extended periods of time.

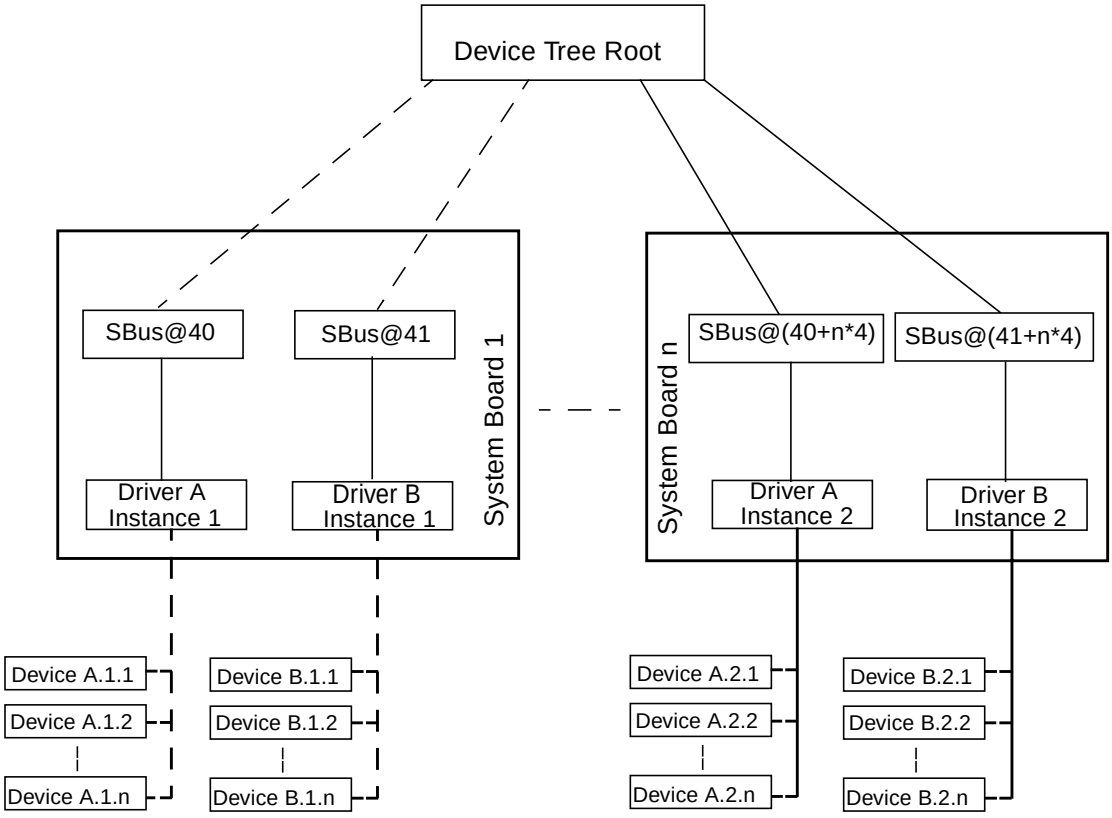
The specific lock in question is the `p_selock` of the `page_t` structure, typically acquired through the `page_lock()` routine. The `p_selock` is intended as a transient lock; holding the lock interferes with the memory drain required when a system board is detached. The `page_create()`, `page_create_va()` and `page_lock()` routines *are not* DDI/DKI compliant interfaces and their use is strongly discouraged.

Device drivers *must* use the standard DDI memory allocation routines for DMA buffers and/or `kmem_alloc(9F)` for normal kernel dynamic storage. Note that memory allocated by `kmem_alloc(9F)` is "locked" (not subject to paging) and will maintain the same physical address across DR operations (i.e., it is non-relocatable).

If a driver module needs to allocate memory directly via `page_create()`, and the objective is to maintain the memory at the same physical address without modification by the operating system, the function should be invoked with the `PG_NORELOC` flag to indicate to the operating system that such pages are "non-relocatable".

I/O Devices' File Path

FIGURE 1 shows a representation of an I/O device tree to better explain the DR attach and detach functions. As an example, the device file `/devices/sbus@54,0/SUNW,fas@0,8800000/sd@2,0:a` represents a SCSI disk registered with a fast/wide SCSI controller located on `SBus0` of System Board 5 (within a Starfire system).



Note: SBus instances are provided in hex, i.e. System Board 3 contains SBus@4c and SBus4d

FIGURE 1 I/O Device Tree Representation

When detaching a system board, the `dr` driver locates the device tree associated with the system board and traverses it from the bottom up to invoke the `detach(9E)` function on every leaf of the tree.

A full configuration of the system peripherals device tree can be obtained by running the `prtconf(1M)` command. In addition, the `dr(1M)` command on the SSP, supports the `drshow(1M)` function (with the IO option) to display all devices associated with each system board, their physical location, associated drivers, device file, and open instances.

Device Driver DR Functions

For a driver to be fully compliant with the DR framework, it must support the following functions:

- `DDI_ATTACH` and `DDI_DETACH`

  These functions provide the ability to attach or detach a particular instance of a driver without impacting other instances already servicing separate devices. For example, to detach system board 1 in FIGURE 1, the `dr_driver` would invoke the `DDI_DETACH` function on each device under both driver A's first instance and driver B's first instance without affecting other driver instances.

  If you attach a system board that is associated with a driver that is not DR-compliant, the DR operation will not fail, but it will not load the appropriate driver either. As a workaround, you can manually load the driver using the `modload(1M)` command. However, if other instances already exist for that driver, you must first use the `modunload(1M)` to unload the existing instances.

  If you attempt to detach a system board that is associated with a driver that is not DR-compliant, the DR operation *will* fail. As a workaround, you can use the "big hammer approach" by manually unloading the driver using the `modunload(1M)` command. `modunload(1M)` requires that all open device instances be closed and, if necessary, it removes any other instances of the driver.

- `DDI_SUSPEND` and `DDI_RESUME`:

  The `DDI_SUSPEND` and `DDI_RESUME` functions make it possible to detach a board that contains the kernel cage. The *kernel cage* is a special data structure (normally contained on a single system board) which controls the dynamic growth of all non-relocatable memory, including the OpenBoot™ PROM (OBP) and kernel memory.

  When detaching a system board that contains the kernel cage, it is necessary to quiesce the operating system to ensure that no I/O or kernel activity takes place while the kernel cage is being relocated. An operating system quiescence involves the suspension of *all* device driver activity, user threads, and kernel threads to avoid corruption of this critical memory.

Conclusion

I/O Device drivers *must* implement the `DDI_DETACH`, `DDI_ATTACH`, `DDI_SUSPEND`, and `DDI_RESUME` functions to seamlessly support the DR framework.

Under the Solaris 2.5.1 and Solaris 2.6 operating environments, Starfire DR contains a hard-coded list of DR-compliant drivers inside the `dr` module. Starfire DR requires newly created DR-compliant drivers to be registered through the `detach_safe_list1` variable in the `/etc/system` file.

Under the Solaris 7 5/99 operating environment, Starfire DR requires newly-created DR-compliant drivers to set the `D_HOTPLUG` bit in the `cb_ops` data structure.

Dynamic Reconfiguration operations can succeed, even when the system board to be attached or detached is associated with drivers that are not DR-compliant. In this situation, however, you must use the `modload(1M)` and `modunload(1M)`commands, as described in this article.

Author's Bio: Enrique Vargas

Enrique brings a wealth of large systems experience to Sun and specializes in high end UNIXofferings including the Enterprise 10000. Enrique came to Sun from Amdahl where he alsofocused on the high end Solaris systems.