The Linux Device Driver Framework for High-Throughput Lossless Data Streaming Applications

K. Vodopivec, E. Breeding, J. Sinclair (Oak Ridge National Laboratory)

Abstract

Many applications in experimental physics facilities require custom hardware solutions to control process parameters or to acquire data at high rates with high integrity. These hardware solutions typically require custom software implementations. The neutron scattering detectors at the Spallation Neutron Source at Oak Ridge National Laboratory transfer custom protocols over optical fiber connected to a PCI Express (PCIe) read-out board. This full-duplex optical channel is also used as a low-latency communication link to the detector electronics for configuration, real-time health monitoring, and fault detection. A dedicated kernel-mode device driver interfaces the PCIe read-out board to the software application. The device driver must be able to sustain data bursts from a pulsed source while acquiring data for long periods of time. The same optical channel is also used as bi-directional low-latency communication link to detector electronics for configuration, real time health monitoring and fault detection. This article presents a Linux device driver design for a low-latency high-throughput application. Specific areas addressed in this article include: kernel driver optimization techniques, real use case benchmarks and the importance of a clean application programming interface (API) for seamless integration into control systems. This framework has been extended beyond neutron data acquisition, thus, making it suitable for new and diverse applications as well as rapid development of field programmable gate array (FPGA) firmware.

Device Driver for Linux Kernel

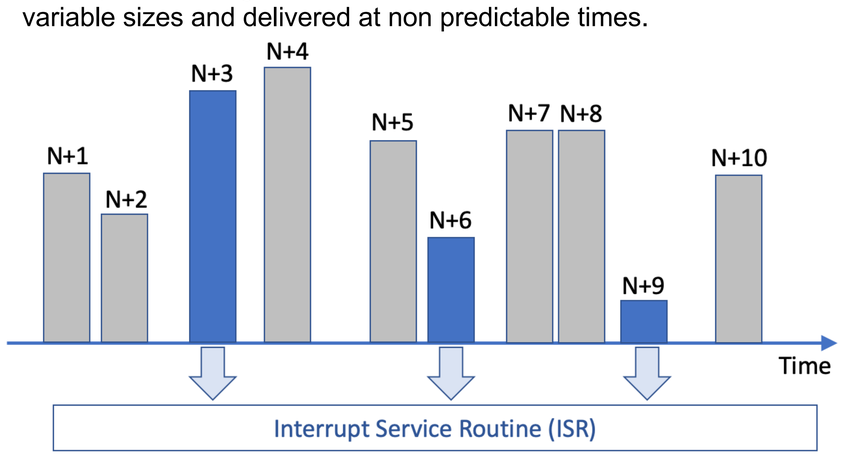
Handle data and fault interrupts
- Packet based protocol triggers system interrupt for each packet.
- Use of interrupt coalescing to effectively reduce number of interrupts.
- Do minimal work to capture DMA state.
- MSI interrupts preferred over legacy interrupts.

Manage memory for DMA
- Fixed physical memory block used as FIFO buffer, PCIe DMA engine pushes data to memory.
- Driver allocates physical memory. By reserving at boot time the contiguous memory can be arbitrary in size.
- Physical memory address is passed to application, data can be processed in place.

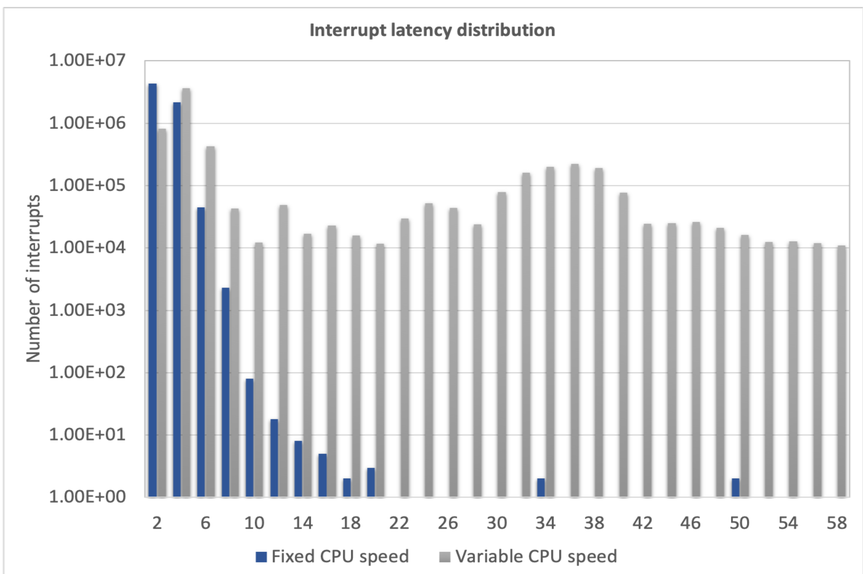
Memory mapped IO
- Allow controlling configuration parameters on FPGA through PCI registers.
- Application manages all registers except ones needed for interrupt handshake.

Interrupt Latency Optimization

Interrupt coalescing decimation number set to 3, packets are of variable sizes and delivered at non predictable times.



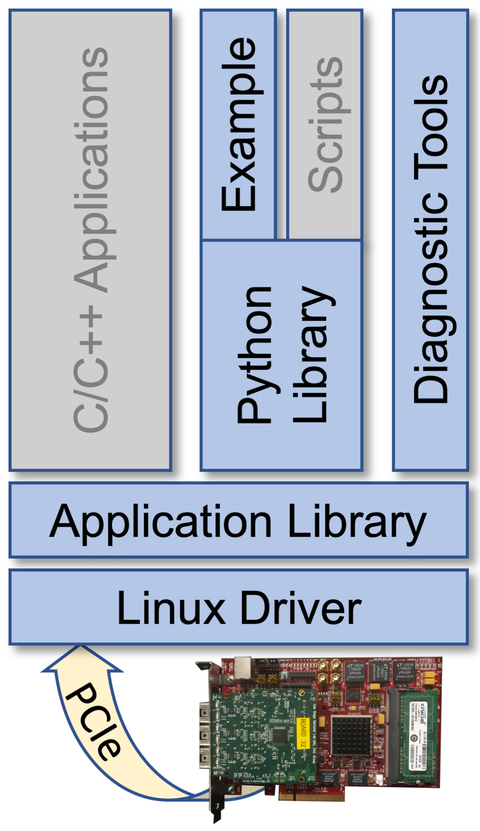
Typical distribution of interrupt latency times on Linux kernel 3.1 with a fixed CPU frequency vs self governing CPU frequency.



Contiguous Memory Allocation

```
BOOT_IMAGE=/vmlinuz-3.10.0 ... memmap=600M$1024M ...

[0.000000] e820: user-defined physical RAM map:
[0.000000] user: [mem 0x0000000000000000-0x000000000009afff] usable
[0.000000] user: [mem 0x000000000009b000-0x000000000009ffff] reserved
[0.000000] user: [mem 0x00000000000e0000-0x00000000000fffff] reserved
[0.000000] user: [mem 0x0000000000100000-0x000000003fffffff] usable
[0.000000] user: [mem 0x0000000040000000-0x00000000657fffff] reserved
[0.000000] user: [mem 0x0000000065800000-0x000000007de1afff] usable
[0.000000] user: [mem 0x000000007de1b000-0x000000007dfd4fff] reserved
[0.000000] user: [mem 0x000000007dfd5000-0x000000007e1ecfff] ACPI NVS
[0.000000] user: [mem 0x000000007e1ed000-0x000000007f351fff] reserved
[0.000000] user: [mem 0x000000007f352000-0x000000007f7fffff] ACPI NVS
[0.000000] user: [mem 0x0000000080000000-0x000000008fffffff] reserved
[0.000000] user: [mem 0x00000000fed1c000-0x00000000fed3ffff] reserved
[0.000000] user: [mem 0x00000000ff000000-0x00000000ffffffff] reserved
[0.000000] user: [mem 0x0000000100000000-0x000000107fffffff] usable
```



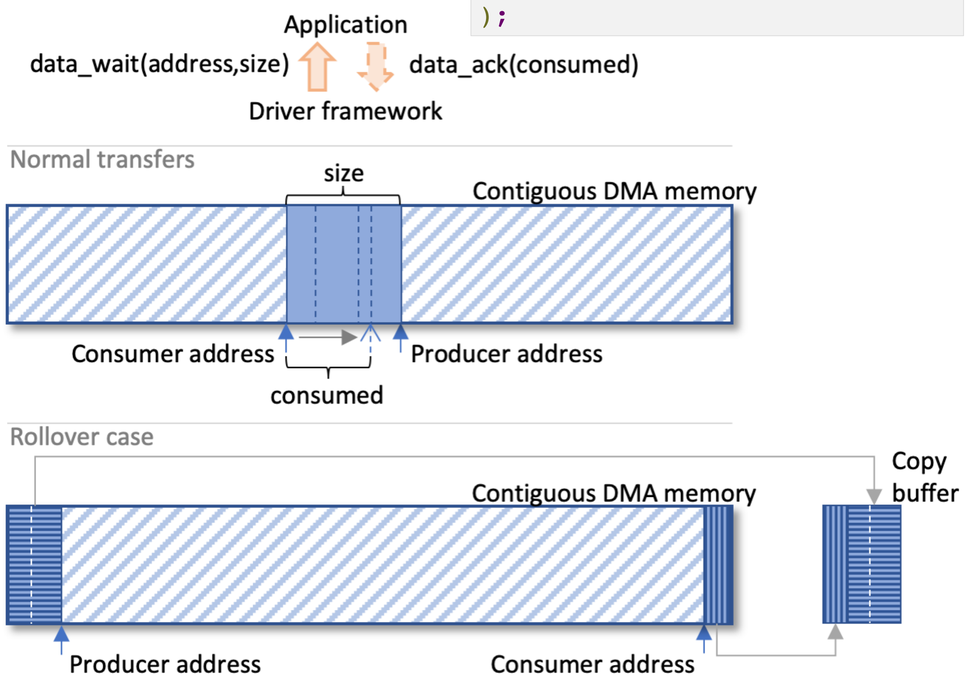
Software Library for C/C++ Applications

Layer between driver and application
- Hides complexity of communicating to device driver from application.
- API is not aware of packet data format

Efficient interfaces for data transfer
- Using zero-copy approach. Ensure safety dealing with physical memory for DMA.
- Handles the FIFO rollover when packet splits.

```
int occ_data_wait(
    struct occ_handle *handle,
    void **address,
    size_t *size,
    uint32_t timeout
);

int occ_data_ack(
    struct occ_handle *handle,
    size_t consumed
);
```



Diagnostic and Prototyping Tools

Application- and data- agnostic tools excel new FPGA firmware functionality. Used for throughput measurements and QA verification before deployment. Powerful troubleshooting tools, help determine cause and source of bugs.

```
$ python
Python 2.7.5 (default, Mar 26 2019, 22:13:06)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux2
Type "help", "copyright", "credits" or "license"
for more information.
>>>
>>> import occ
>>>
>>> device = occ.open("/dev/occ2")
>>> device.enable_rx()
>>>
>>> data = device.read(12)
>>> data
'\x00\x01`\x10\x80\x00\x00\x00\x1d\x00\x00\x00'
>>>
>>> device.send(data)
>>>
>>> for addr in range(0, 20, 4):
...   print hex(device.io_read(0, addr))
0xb0002
0x8001a
0xcc76648
0xffffffff
0xffffffff
>>>
>>> device.io_write(0, 0x84, 0xF)
```


Application Use Cases

- Event-based data acquisition at 17 neutron scattering instruments at Spallation Neutron Source and 4 at High Flux Isotope Reactor, Oak Ridge National Laboratory
- Fast data acquisition of raw amplitude and phase data for LLRF APS Upgrade project, Argonne National Laboratory
- Supported rapid development of FPGA firmware for Injection Kicker Waveform monitor and the Machine Protection system controller for Power Upgrade project at Spallation Neutron Source

SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S.