UDI Technology Benefits

Introduction

- **UDI design goals**
  - **Cross-OS portability**
  - **High performance and scalability**
- **OS-neutrality benefits beyond portability**
- **Technology benefits even for single OS**

OS-Neutrality

- **Precise separation between device semantics and OS semantics**
- **Flexibility to evolve OS without re-basing drivers**
- **Implementation flexibility allows OS value-add**

Benefits of UDI Driver Model

- Direct context pointer on entry
- Automatic endian conversions
- Scatter/gather lists built by OS
- Implicit synchronization
- Non-blocking execution model
- Dynamic configuration (hot plug)
- Location independence
- Tracing & logging

Direct Context Pointer on Entry

- **Each entry point is passed a context pointer (void *)**
- **Context pointer previously set by driver**
- **Allows direct access to per-instance or per-request data structures**
- **Simpler and faster**

Automatic Endian Conversions

- Driver knows the endianness of its device but not of intervening hardware
- Some systems have hardware byte swap
- Driver simply specifies device endianness
- Environment byte swaps if necessary
- No overhead in non-swap case

Scatter/gather Lists Built by OS

- **Scatter/gather list specifies array of bus address, length pairs for DMA engine to locate data**
- **DMA drivers receive scatter/gather lists pre-built according to their needs**
- **Compatible with IEEE 1212.1**
- **Many drivers will just pass address of list to their device**

Implicit Synchronization

- **No MP locking primitives or interrupt masking in driver code**
- **However, region model allows driver instances to be fully parallelized and preemptable**
- **Default granularity is per device**
- **Optional fine-grained parallelism by spawning additional regions**

Non-blocking Execution Model

- **Environment service calls use completion callbacks rather than blocking**
- **Optimized for immediate callbacks**
- **Gives OS fine control over scheduling driver activity and resources**
- **Keeps driver execution model simple**
- **Requires less memory when waiting for resources (no thread stacks)**

Dynamic Configuration (Hot Plug)

- **Instance independence allows new devices to be bound to drivers at any time**
- **Full hot-plug support through device management entry point**
  - **Suspend, Resume**
  - **Replace, Remove**
- **Also supports power management**

Location Independence

- **UDI environments can support a wide range of architectural models**
- **UDI drivers can be run in user mode or kernel mode**
- **UDI drivers can be run in global or private address spaces**
- **UDI drivers can run on host CPUs or I/O processors**

Tracing & Logging

- **Central error logging**
  - **Error correlation allows errors to be tied to root cause**
- **Dynamic run-time tracing**
  - **Informational logging**
  - **Individual message categories can be turned on and off at any time**

Conclusion

- **UDI is not just for portability**
- **Advanced technology allows driver simplicity and OS flexibility**
- **Portability doesn't have to mean lower performance**