Datacenter Fabric Workshop

Windows IB

Windows Core SW
User Mode Future

Fab Tillier

SilverStorm Technologies

ftillier@silverstorm.com

August 22, 2005

Agenda

- Overview
- HW Resources
- Hardware Events
- Completion Events

Overview

- Evolve API to more closely resemble Win32 API model
  - API functions return BOOL or pointers
  - Use Set/GetLastError() for detailed status
- Reduce learning curve for clients
- Take advantage of Win32 I/O notification mechanisms and semantics
  - Reduce number of threads in processes
  - First step in supporting single threaded apps

Agenda

- Overview
- HW Resources
- Hardware Events
- Completion Events

HW Resources

- Public structures instead of opaque handles
  - Similar to OpenIB Gen2 Linux
  - Contain attributes for use by user (read only)
  - Provides vtable for operations
  - Reduces duplication of information
  - Reduces function parameters
  - Eliminate query functions

HW Resources

```
typedef struct _IB_MR
{
    IB_MR_OPS                  *vtbl;

    IB_PD                      *pPd;
    UINT8* __ptr64             pLocalStart;
    UINT8* __ptr64             pLocalEnd;
    UINT8* __ptr64             pRemoteStart;
    UINT8* __ptr64             pRemoteEnd;
    DWORD                      desiredAccess;
    UINT32                     lKey;
    UINT32                     rKey;

}   IB_MR;
```

No need for user to duplicate LKey or RKey

VTable Example

```
struct _IB_PD_OPS
{
    IB_MR* (*RegisterMr)(
           IB_PD *pPd,
           IB_MR_CREATE *pMrCreateAttr );
};


struct _IB_MR_OPS
{
    BOOL (*Deregister)(
           IB_MR *pMr );

    BOOL (*Modify)(
           IB_MR *pMr,
           DWORD MrModMask,
           IB_PD *pPd,
           IB_MR_CREATE *pMrCreateAttr );
};
```

No need to return LKey or RKey

Agenda

- Overview
- HW Resources
- Hardware Events
- Completion Events

Hardware Events Goals

- Provide file semantics for affiliated and unaffiliated events
- Allow the client to decide how to get notifications
  - Synchronous
  - I/O Completion Ports
  - Asynchronous Procedure Call
  - GetOverlappedResult

Hardware Events Operation

- Single completion model per open HCA
  - HCA, CQ, QP events all use the same file handle
- Cannot use APCs if using I/O Completion Ports
  - Same limitations as documented for CreateIoCompletionPort
- File created on demand
  - No need to create file if no events are ever requested

CA Events

```
IB_CA* OpenCa(
    IN    UINT64 guid,
    IN    DWORD flags );


struct _IB_CA_OPS
{
    …
    HANDLE (*GetAsyncFile(
           IB_CA *pCa );

    BOOL (*GetAsyncEvent)(
           IB_CA *pCa,
           BOOL affiliated,
           IB_ASYNC_EVENT *pAsyncEvent,
           OVERLAPPED *lpOverlapped,
           OVERLAPPED_COMPLETION_ROUTINE *lpCompletionRoutine );
    …
};
```

FILE_FLAG_OVERLAPPED for asynchronous event reporting

Get asynchronous event file handle to bind to I/O completion port

Get CA affiliated or unaffiliated event

CQ & QP Affiliated Events

```
struct _IB_CQ_OPS
{
    …

    BOOL (*GetAsyncEvent)(
           IB_CQ *pCq,
           IB_ASYNC_EVENT *pAsyncEvent,
           OVERLAPPED *lpOverlapped,
           OVERLAPPED_COMPLETION_ROUTINE *lpCompletionRoutine );
};
```

Get CQ affiliated event

```
struct _IB_QP_OPS
{
    …

    BOOL (*GetAsyncEvent)(
           IB_QP *pQp,
           IB_ASYNC_EVENT *pAsyncEvent,
           OVERLAPPED *lpOverlapped,
           OVERLAPPED_COMPLETION_ROUTINE *lpCompletionRoutine );
};
```

Get QP affiliated event

Agenda

- Overview
- HW Resources
- Hardware Events
- Completion Events

Completion Events Goals

- Provide file semantics for completion events
- Allow the client to decide how to get notifications
  - Synchronous
  - I/O Completion Ports
  - Asynchronous Procedure Call
  - GetOverlappedResult

CQ Events Operation

- Available on a per-CQ basis
  - Allows a unique key value when binding to I/O completion port
  - Requires a file object per CQ
- Cannot use APCs if using I/O Completion Ports
  - Same limitations as documented for CreateIoCompletionPort
- File created on demand
  - No need to create file if user never calls Rearm

CQ Events

```
IB_CQ* (*CreateCq)(
   IB_CA *pCa,
   SIZE_T nCqe,
   DWORD flags );
```

FILE_FLAG_OVERLAPPED for asynchronous completion events

```
struct _IB_CQ_OPS
{
   …
   HANDLE (*GetCompFile(
        IB_CQ *pCq );

   BOOL (*Rearm)(
        IB_CQ *pCq,
        DWORD notifyType,
        OVERLAPPED *lpOverlapped,
        OVERLAPPED_COMPLETION_ROUTINE *lpCompletionRoutine );
   …
};
```

Get completion event file handle to bind to I/O completion port

Get CA affiliated or unaffiliated event

Completion Events Options

- Single per open HCA CQ event file handle
  – Fewer file objects in application
  – No per-CQ key when binding to I/O completion port
  – Separate from affiliated and unaffiliated events file handle
  – Use OVERLAPPED to distinguish I/O operations
  – No mixing I/O Completion Ports and APCs

Resources

- OpenIB WiKi
  - https://openib.org/tiki/tiki-index.php?page=OpenIB+Windows
- Openib-windows mailing list
  - http://openib.org/mailman/listinfo/openib-windows
- Sign up to contribute
  - http://windows.openib.org/openib/contribute.aspx

Q & A