

# Solaris Open Fabrics User Verbs Architecture Document

Editor	Pramod Gunjekar
Authors	Pramod Gunjekar Brendan Doyle
Date	07/11/07
Version	0.3
Last Modified by	<a href="mailto:pramod.gunjekar@sun.com">pramod.gunjekar@sun.com</a> brendan.doyle@sun.com
File name	solaris_ofuv_arch

## Table of Contents

1	Revision History.....	3
1.1	References.....	3
2	Glossary.....	4
3	Introduction.....	5
4	Open Fabrics Enterprise Distribution Architecture .....	6
4.1	Components.....	6
4.2	OF User Level Verbs API.....	7
4	Solaris OF-UV Overview.....	10
4.1	Solaris OF-UV Goals and Non Goals.....	10
4.1.1	Goals .....	10
4.1.2	Non-Goals.....	10
4.2	Solaris OF-UV Project Scope.....	10
4.3	Risks and Assumptions.....	12
5	Solaris OF-UV Architectural Considerations.....	13
6	Solaris OF-UV Phase-I .....	17
6.1	Architecture.....	17
6.2	OF-UV Library Path.....	19
6.3	Discovery of HW providing Verbs functionality.....	19
6.4	Loading of Userland Drivers.....	19
6.5	Solaris extensions to libibverbs.....	19
6.6	Solaris librdmacm architecture.....	19
6.7	Solaris IBTF/HCA driver modifications.....	20
7	Solaris OF-UV Phase-II .....	21
7.1	Architecture.....	21
7.2	iWARP Support.....	21
7.3	librdmacm Support.....	22
7.4	Support for Mellanox Arbel and Hermon Controllers.....	23
7.5	Support for future IB HCAs.....	23
7.6	SA Access and User MAD library.....	23
8	Solaris OF-UV test development and testing.....	24

## 1 Revision History

<i>Revision</i>	<i>Date</i>	<i>Summary</i>
0.1	03/16/07	Initial Draft Version.
0.2	03/22/07	After internal review
0.3	07/11/07	Revised Phase-I Architecture

### 1.1 References

*OpenIB Architectural Overview – Roland Dreier and others*

*OpenFabrics Software Stack – June 2006*

*OFED Update – Presentation in IBTA-OFA presentation on Sept. 2006*

*UDAPL Porting Guide, version 8.11*

*IBTF Architecture Document*

*Open MPI code base at : <http://www.open-mpi.org/>*

## 2 Glossary

<b>OFED</b>	Open Fabrics Enterprise Distribution
<b>OF-UV</b>	Open Fabrics User Verbs
<b>ULP</b>	Upper Layer Protocol
<b>BTL</b>	Byte Transport Layer
<b>uDAPL</b>	User Direct Access Programming Library
<b>MAD</b>	Management Datagram
<b>DR</b>	Dynamic Re-configuration
<b>IBTF</b>	Infiniband Transport Framework
<b>IBTF VTI</b>	IBTF Verbs Transport Interface

### 3 Introduction

The Open Fabrics Enterprise Distribution (OFED) supports InfiniBand and iWARP fabrics. OFED is developed from OpenFabrics alliance ([www.openfabrics.org](http://www.openfabrics.org)), which is a collaborative open source development. OFED is currently supported on Linux environments.

The OFED distribution consists of multiple components. The Solaris Open Fabrics User Verbs API (OF-UV) is a key component of OFED. The Open Fabrics User Level Verbs API provides support for user level ULPs like MPI, Cluster File Systems, Open SM and user level Diagnostics utilities. The Open Fabrics User Verbs libraries are also available in Linux distributions like Debian, Ubuntu and Fedora.

The distributed file system, Lustre, from Cluster File System Inc., uses the Open Fabrics Verbs libraries for IB transport. Support for OF-UV on Solaris, is required for supporting Lustre over InfiniBand, on Solaris.

The Open-MPI source code uses OF-UV interfaces for IB transport functionality. The current Solaris Open MPI has been implemented using a User Direct Access Programming Library (uDAPL) Byte Transport Layer (BTL), as Solaris OF-UV support is not currently available. There is a large community effort developing Open MPI using the Open Fabrics User Level Verbs API. The Solaris Open MPI implementation can leverage these community efforts better, by adopting to Solaris OF-UV implementation in the future. The Open MPI implementation using OF-UV is expected to be more efficient than the implementation using uDAPL.

Lustre and Open MPI are the initial potential consumers of Solaris OF-UV interfaces. Subnet Manager related diagnostic tools are also planning to use OF-UV interfaces in the future. Solaris OF-UV support will also be an enabler for porting ULPs and tools available within OFED and in other distributions, to Solaris.

This document defines the OF-UV architecture and details the functional, development and testing requirements of OF-UV support on Solaris.

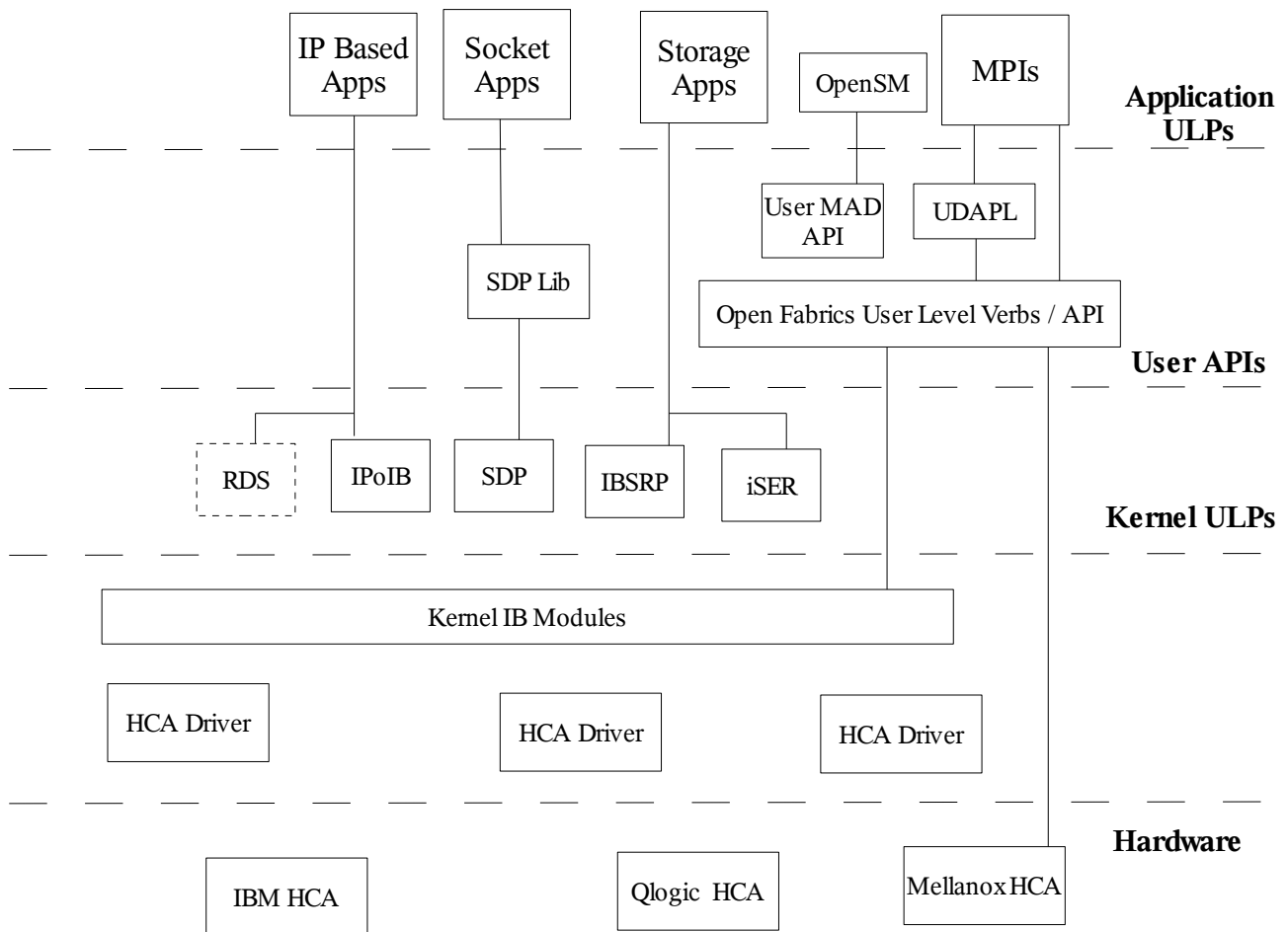
## 4 Open Fabrics Enterprise Distribution Architecture

### 4.1 Components

The OFED Linux implementation consists of multiple components:

1. Various MPIs - Open MPI, MPICH
2. Upper Layer Protocols – SRP, SDP, RDS , iSER, uDAPL
3. Open Fabrics User Level Verbs / API (OF-UV)
4. User Level MAD API
5. User level SDP library
6. Open SM
7. Test and Performance Tools

The general architecture of OFED stack is as represented in the diagram below.



## 4.2 OF User Level Verbs API

The OF User Level Verbs API provides support to Application level ULPs like MPI and Lustre. The OF-UV interfaces provide support for Infiniband verbs (RC, UD & UC transport services are supported). Additionally libraries for supporting communication management, MAD transport and SA Access are also provided. The components of OF User Level library interfaces, that are of interest to this project are :

1. Verbs library - *libibverbs*
2. User level HCA Drivers
  - ◆ User level Driver for Mellanox HCAs - *libmthca*
3. Communication Management library
  - ◆ *librdmacm*
4. MAD transport library and SA Access library
5. Support for transport services : RC/UD

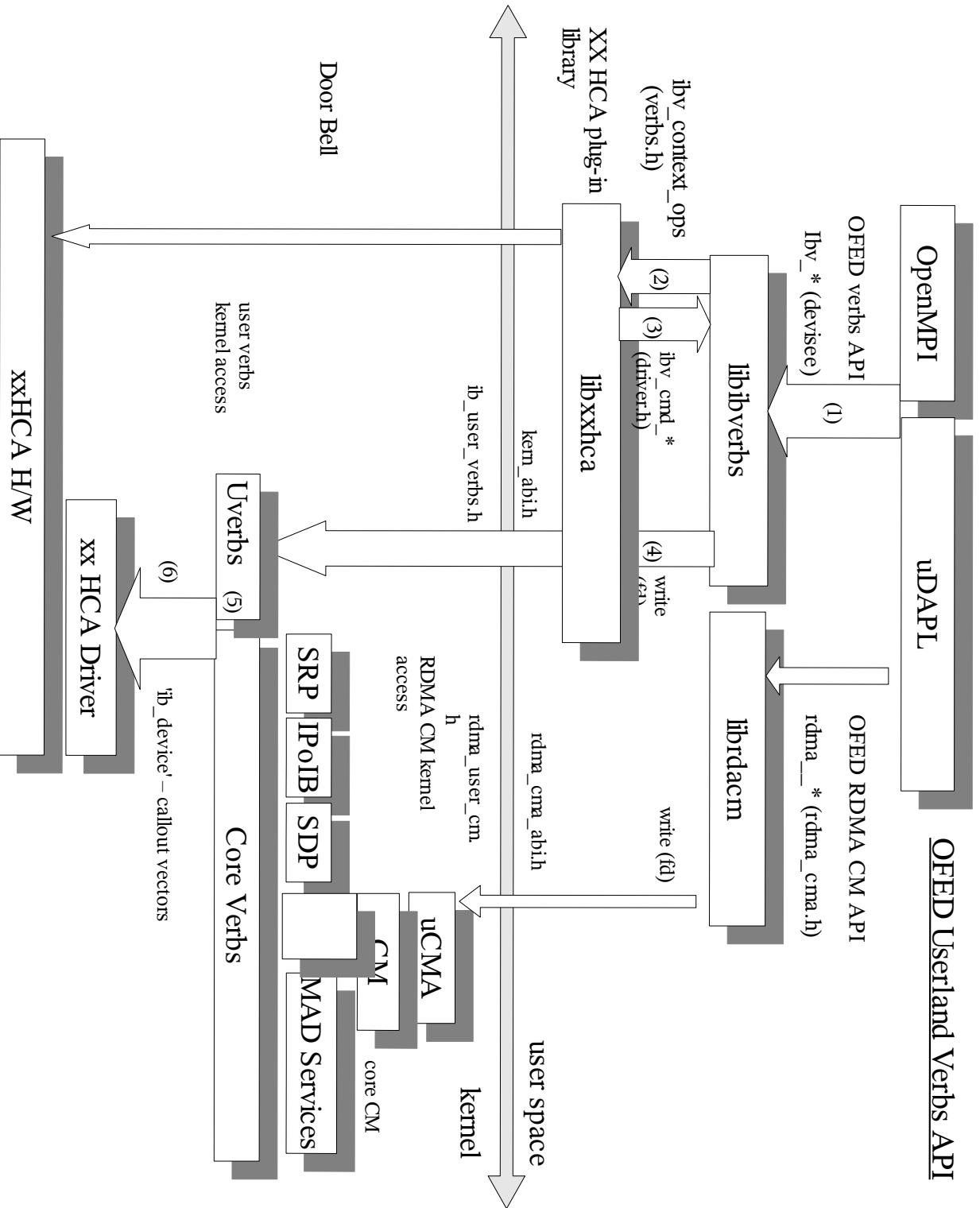
The Verbs library provides an API which maps to the IB Verbs, as specified in Volume 1 (not all verbs are supported, only those that are version 1.1 compliant are supported). Application level ULPs like Open MPI and Lustre use the Verbs library for communicating over IB. The user level HCA drivers are plug-in libraries which provide Verbs support for a class of HCAs. The OFED libraries on Linux provide support for Mellanox, Pathscale/QLogic and IBM HCAs. This project will support Mellanox HCAs only. All other HCAs are out of scope of this project.

The *librdmacm* library provides connection management functionality. This library provides a generic RDMA set of CM interfaces that can run over iWARP and IB. *librdmacm* also provides abstracted SA access functionality in “find path” APIs. The UDAPL implementation in OFED uses the *librdmacm* library functionality. The *libibcm* library also provides API that support for ULPs for connection establishment. The *librdmacm* is expected to replace *libibcm*. Support for *libibcm* is not of interest for this project.

The SA access library provides interfaces for querying IB subnet information. MPICH implementations is the only current user of the SA access functionality. Currently the only SA access functionality which is supported by OFED 1.0 is the ability to retrieve PathRecords and manage multicast groups.

OFED user land libraries support transport services for: Reliable Connected(RC), Unreliable Datagram (UD) and Unreliable Connect (UC). There is no known need for exposing the Unreliable Datagram to user level, at this point of time.

The diagram overleaf represents the OF-UV architecture and interfaces on Linux.



The API flow is:

1. Application makes `ibv_*` API call into `libibverbs`
2. `libibverbs` calls into HCA library via `ibv_context_ops` vector
3. HCA library performs HCA specific operations and calls back into `libibverbs` via `ibv_cmd_*` API
4. `libibverbs` sends `kern_abi.h` cmd to uVerbs kernel access module via write on fd
5. uVerbs calls HCA userland specific entry point via `ib_device` vector
6. HCA userland entry point performs userland specific operations and calls core HCA entry point

Create QP example:

1. Application calls `ibv_create_qp()`
2. `ibv_create_qp()` calls `mt_hca_create_qp()` via `ibv_context_ops.create_qp()` vector
3. MT HCA library performs MT specific operations and calls back into `libibverbs` via `ibv_cmd_create_qp()`
4. `libibverbs` calls into uVerbs kernel entry point `ib_uverbs_create_qp()` via `kern_abi.h` cmd write on fd
5. `ib_uverbs_create_qp()` calls `mt_hca_create_qp()` via `ib_device.create_qp()` vector
6. `mt_hca_create_qp()` performs userland specific operations and calls `mt_hca_alloc_qp()`

## 4 Solaris OF-UV Overview

### 4.1 Solaris OF-UV Goals and Non Goals

#### 4.1.1 Goals

1. Support all OF-UV APIs exposed to ULPs, conforming to OFED header files. Linux specific components (like sysfs related parameters) may either be emulated or have Solaris equivalent structures.
2. No degradation of latency and bandwidth with respect to Linux OFED implementation. For example, the number of copies during I/O has to be equal to or less than the OFED Linux implementation.
3. Add Solaris specific enhancements for supporting features such as , DR, etc.
4. Optimal use of the existing Solaris IB code.
5. Seamless integration of other Open Fabrics APIs from Linux.
6. The Solaris OF-UV architecture should be open to accommodate other IB HCAs like Pathscale and iWARP controllers.
7. The performance of Solaris OF-UV will be equal to or better than that of Solaris UDAPL implementation. The performance of Open MPI using UDAPL BTL and OF-UV BTL will be the basis for performance comparison.
8. This project will support IB verbs, as specified in InfiniBand Architecture Specification Volume 1, Release 1.1, as supported by OFED 1.2.

#### 4.1.2 Non-Goals

1. No putbacks/bug fixes to OFED. Most parts of the Solaris implementation of OF-UV will to be very different from the Linux implementation, consequently putbacks to OFED will not be relevant.
2. Support and testing for ULPs other than Open MPI and Lustre (not guaranteed).
3. No plan for supporting *libibcm*. *libibcm* is planned to be depreciated.
4. Support for Open Fabrics Subnet Access and User MAD libraries is out of scope of this project. The support for these libraries can be provided by projects, which are independent of this project.

### 4.2 Solaris OF-UV Project Scope

Solaris OF-UV will include all interfaces required for Application level IB ULPs. Solaris specific requirements like DR support, will also be added. The project plans to provide interfaces required for

supporting Open MPI and Lustre. The details are given below:

<i>OF-UV Component</i>	<i>Solaris Support</i>	<i>Comments</i>
<b>libibverbs</b>	Supported by the project	Required by Open MPI and Lustre
<b>libmthca</b>	Supported by the project	Required by Open MPI and Lustre
<b>RC</b>	Supported by the project	Required by Open MPI and Lustre. ULPs can use Out Of Band protocols (as in Open MPI) or <i>librdmacm</i> to obtain the remote information required for establishing RC connection.
<b>librdmacm</b>	Support by the project	Required by Lustre
<b>UD</b>	Support can be a follow up of this project	Open MPI or Lustre do not use UD. UD support is part of libibverbs. Enabling UD support can be taken as a follow-on to this project.
<i>libibpathverbs</i> , <i>libehca</i> and other HCAs	Support when Solaris supports additional HCAs.	Support for OF-UV user plug-in library can be part of the overall support for a new HCA on Solaris.
SA access libraries	It is not known if this support will be required. Any project to support SA access libraries on Solaris should be independent of this project.	SA access library ( <i>osmv_query_sa()</i> and related calls) are used by MPICH and some diagnostic tools. There is no requirement for MPICH support on Solaris.  It is not known if this support will be required in the future
User MAD library	Any project to support user MAD libraries on Solaris should be independent of this project.	Diagnostic tools, SM utilities use the MAD library. There is a potential requirement for User MAD library support in the future.
<b>UC</b>	The requirement for UC at user level, can be re-evaluated in the future.	Solaris IB kernel infrastructure currently does not support Unreliable connect on IB fabric. There is currently no known requirement for UC support in the user level. This can be re-evaluated in the future.
<b>libibcm</b>	Not required	OFED is migrating from <i>libibcm</i> to <i>librdmacm</i> for providing connection management functionality for ULPs.

### **4.3 Risks and Assumptions**

1. Any requirement for support for ULPs other than Open MPI and Lustre can change the requirements.
2. The requirements are based on the current understanding of Open MPI and inputs from CFS/SFW. Any change in the requirements will effect the project.
3. No major changes are expected in the existing Solaris IB software stack for the phase-I of the project. Any requirement for major change will affect this project.
4. Arbel should provide support for OS-bypass for UDAPL. This support should be similar to tavor driver support for OS-bypass.

## 5 Solaris OF-UV Architectural Considerations

The Solaris OF-UV implementation will support interfaces defined in the OFED *verbs.h* header, and exported by the OFED *libibverbs* library. Linux specific components (like *sysfs* related parameters) may either be emulated or have Solaris equivalent structures.

HCA User Level plug-ins use the following interfaces to interact with the kernel :

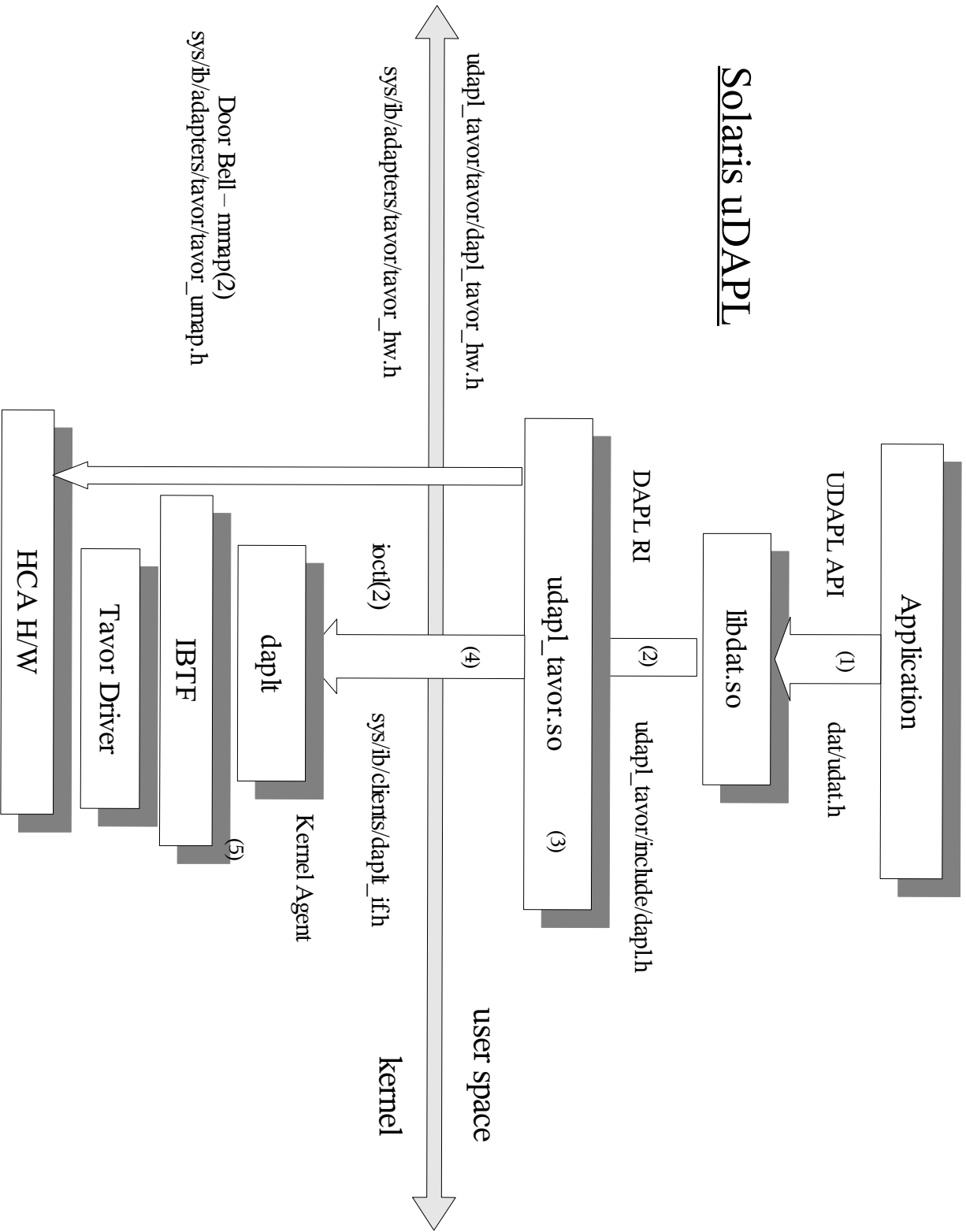
- Interface with a Solaris kernel agent supporting OF APIs
- Interface to HCA hardware for Kernel bypass/fastpath

The existing Solaris uDAPL implementation has similar components, and provides kernel bypass support for Mellanox *tavor* HCAs (and *Arbel* controllers in *tavor* mode). The interfaces defined in *dapl\_tavor\_hw.h/tavor\_umap.h* provide the kernel bypass mechanism for the uDAPL *tavor* plug-in. The *tavor(7d)* driver contains the following enhancements to support OS-bypass for uDAPL:

- *mmapping* the UAR doorbell page for synchronizing with the HCA hardware
- *mmapping* CQs for OS bypass polling of work request completions
- *mmapping* QPs for OS bypass posting of work requests
- Locking down of user memory during MR registration.

The architecture and implementation of which is shown diagram overleaf:

# Solaris uDAPL



The API flow for the UDAPL implementation is:

1. Application makes DAT call into libdat.so library
2. libdat.so calls into hca specific library using `dapl_*` APIs
3. `dapl_*` APIs call IB specific `dapls_ib_*` APIs
4. `dapls_ib_*` APIs call into kernel `daplt` using `daplt_if.h` ioctl cmds
5. `Daplt` interfaces to HCA driver using IBTF APIs

Example:

1. Application calls `dat_ep_create()`
2. `dat_ep_create()` calls `dapl_ep_create()` via DAT provider vector
3. `dapl_ep_create()` calls `dapls_ib_qp_alloc()`
4. `dapls_ib_qp_alloc()` calls into kernel `daplt` `daplka_ep_create()`, using `daplt_if.h` `EP_CREATE` ioctl cmd
5. `daplt` `daplka_ep_create()` calls IBTF `ibt_alloc_rc_channel()`

The interface with the Solaris kernel agent supporting OFED, should support all verbs, which includes support for non-protected verbs (for HCAs which do not support kernel bypass for non-protected mode verbs). The interface with the kernel agent for OFED can be either be based on OFED interfaces as defined in `kernel_abi.h` and `ib_user_verbs.h` or uDAPL interfaces as defined in `daplt_if.h`. The benefits of adopting the OFED interfaces are :

- Easier portability for future User level drivers
- Easier future iWARP support .
- Preferred long term architecture for Solaris OF stack

The benefits of adopting to modified uDAPL kernel agent interface are :

- Quicker implementation path
- Optimal use of the existing Solaris IB stack

The uDAPL kernel agent interface `daplt_if.h` provides interfaces to support uDAPL style semantics; End Point(EP) etc, basic Verbs operations for allocating a QPs etc are not defined. To use this option the interface needs to be extended and modified to support the APIs defined in `libibverbs`.

Given the time constraints the first phase of Solaris OF-UV will be based on the uDAPL plug-in library and the uDAPL kernel access interface. In the second phase the interfaces will be migrated to OFED style of interfaces.

Linux OF-UV uses read/write system calls to interface with the kernel agent. Solaris UDAPL uses the

ioctl(3) interface for UDAPL style interfaces. The relative performance benefits of using read(3), write(3) system calls versus ioctl(3) on Solaris, are unknown at this point. For Phase-I the uDAPL ioctl(3) interfaces will be used. For Phase-II, a choice will to be made between ioctl(3) and read(3)/write(3) based on measurements of performance with alternative implementations.

## 6 Solaris OF-UV Phase-I

### 6.1 Architecture

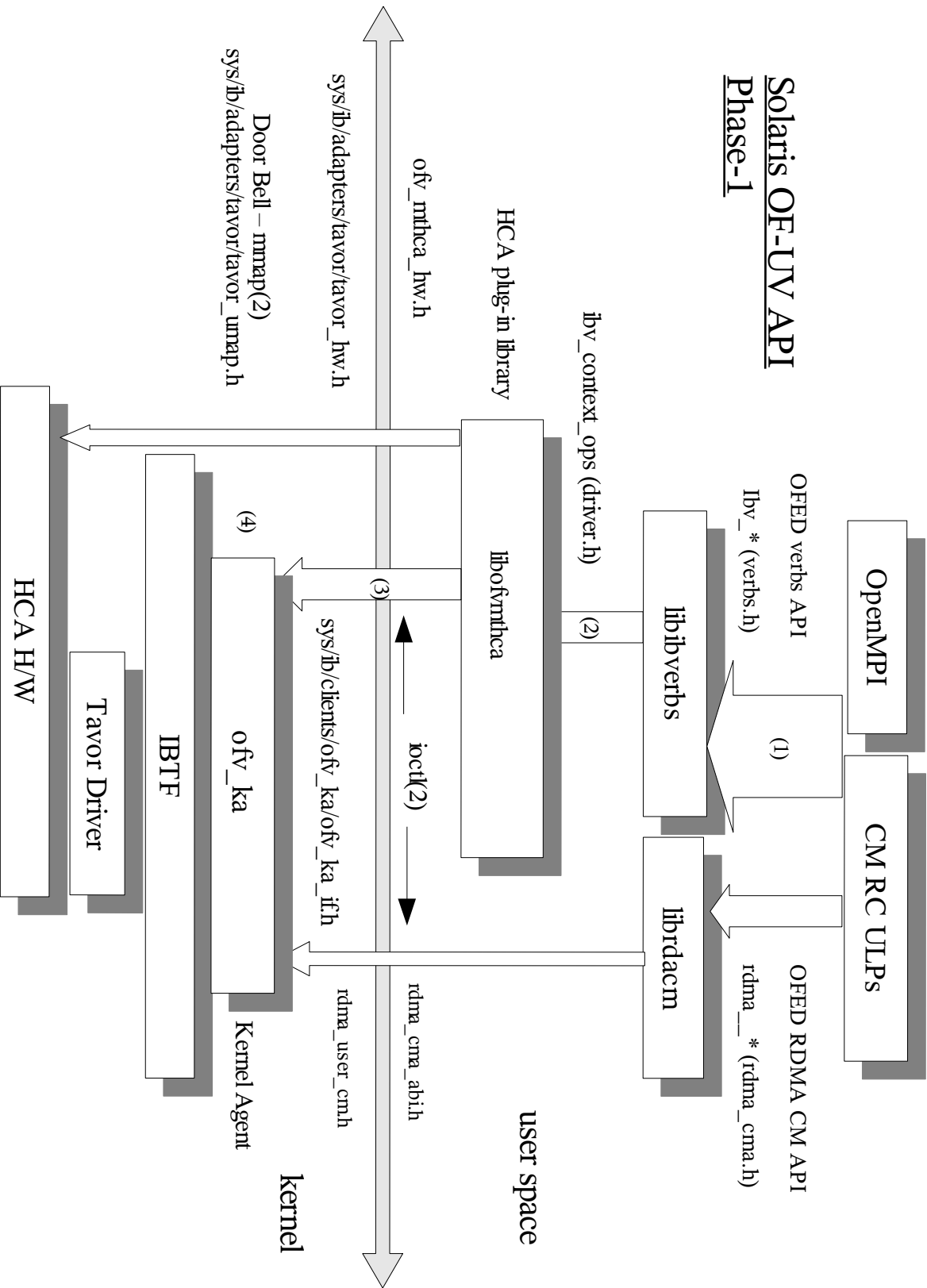
The phase 1 architecture/implementation is based on porting *libibverbs* from OFED, and cloning *tavor\_udapl.so* and *daplt* from uDAPL.

The Solaris OF-UV HCA plug-in library (*libofvmthca*) is based on the uDAPL *tavor\_udapl.so* plug-in, *tavor\_udapl.so* is copied and renamed, then modified to interface into *libibverbs* using the interfaces defined in the OFED *verbs.h* header. Calls are made from *libibverbs* into the Solaris OF-UV HCA plug-in library via the OFED defined *ibv\_context\_ops* vectors, however unlike OFED *libofvmthca* will not call the *ibv\_cmd\_\**() interfaces (defined in *driver.h*) to call back into *libibverbs*, instead it will call directly into the Solaris OF-UV kernel agent via the uDAPL style ioctls.

The *libofvmthca* plug-in will use a modified uDAPL *daplt\_ih.h* to interface to the modified uDAPL *daplt* kernel access module. uDAPL specific structures EPs, etc will be removed and OFED verbs specific structure QPs etc will be added.

The Solaris OF-UV kernel access agent (*ofv\_ka*) provides the interfaces for supporting OF-UV and is based on the *daplt* IBTF Pseudo client. The kernel agent interfaces with the userland HCA plug-in library using modified DAPLT interfaces. The kernel agent uses the IBTF VTI interfaces for verbs related functionality. The kernel bypass interface will be the same interface used by uDAPL.

The figure overleaf gives an overview of the Solaris OF-UV phase-I architecture :



The API flow is:

1. Application makes `ibv_*` API call into *libibverbs*
2. *libibverbs* calls into HCA plug-in library (*libofvmtbca*) via `ibv_context_ops` vector
3. *libofvmtbca* calls into the kernel access *ofv\_ka* driver using the modified *daplt\_if.h* IOCTL commands.
4. The *ofv\_ka* driver interfaces to the HCA driver using the IBTF VTI interfaces.

## 6.2 OF-UV Library Path

The libraries *libibverbs* and *librdmacm* will be located in the `/usr/lib/ofuv` directory. The plug-in drivers will be in the `/usr/lib/ofuv/hw` directory, by default. The typedef for the userland initialization function pointer *ibv\_driver\_init\_func()* defined in *driver.h* will be used as in Linux OFED.

## 6.3 Discovery of HW providing Verbs functionality

The kernel agent providing the Verbs functionality will provide an IOCTL, which will return the information related to all HW controllers providing Verbs functionality.

## 6.4 Loading of Userland Drivers

The *libibverbs* library will load userland drivers from the `/usr/lib/ofuv/hw` directory, by default. The library will also use the path specified by the calling user's environment, using the environment variable, `OPENFABRICS_DRIVER_PATH_ENV`. This is similar to the OFED implementation on Linux.

## 6.5 Solaris extensions to libibverbs

The OF-UV does not completely support Solaris specific features like HCA DR and memory DR . The Solaris OF-UV project will define methods for ULPs to support these Solaris specific features. These methods can be based on the existing Solaris support like RCM (Re-Configuration Manager) or extensions to the OF-UV interfaces. The details of the Solaris specific extensions have not been finalized.

The Solaris specific extensions will be designed in such a manner, that it is independent of the OF-UV interfaces. The ULPs should be able to support the Solaris specific extensions with zero or very minimal changes in the ULP code using the OF-UV interfaces (defined in *verbs.h*). The Solaris specific extensions will be defined in separate header file(s).

For the Open MPI/Lustre implementation, the intent is to just add source code to support the Solaris specific extensions, with no or very little modifications to the Open MPI BTL/Lustre sources. Details of achieving the same have to be finalized.

## 6.6 Solaris *librdmacm* architecture

The Solaris OF-UV *librdmacm* will provide interfaces as defined in the OFED *rdma\_cma.h* header to

application ULPs. The *librdmacm* library will interface with the OF-UV kernel access agent (*ofv\_ka*) using interfaces as defined by the OFED *rdma\_cma\_abi.h* header. No separate *ucma* kernel driver will be supported in the Phase-I of Solaris OF-UV support. The OF-UV kernel access agent, will use the IBTF IP address related APIs to support the RDMA CM related functionality.

## 6.7 Solaris IBTF/HCA driver modifications

The Solaris IB Transport framework and HCA drivers may require minor modifications for supporting OF Verbs. The intent is to keep the changes minimal. However IBTF changes may be required for functionality or for better performance of OF Verbs.

One potential IBTF change that is desirable is to provide interfaces for posting Work Requests as a linked list. The current IBTF implementation provide interfaces to post an array of Work Requests, whereas Open Fabrics User Verbs/API provide support for posting a linked list of Work Requests. Extending IBTF and HCA drivers to provide an interface to post a linked list of work requests will be reduce unnecessary WR processing overhead..

This project will also use the “Get IP address” IBTF interfaces for emulating the RDMA CM functionality.

The list of potential changes to IBTF is not final. The OF-UV interfaces and the Solaris IBTF interfaces have to be mapped and any change required for functionality and better performance needs to listed. This is currently work in progress.

## 7 Solaris OF-UV Phase-II

### 7.1 Architecture

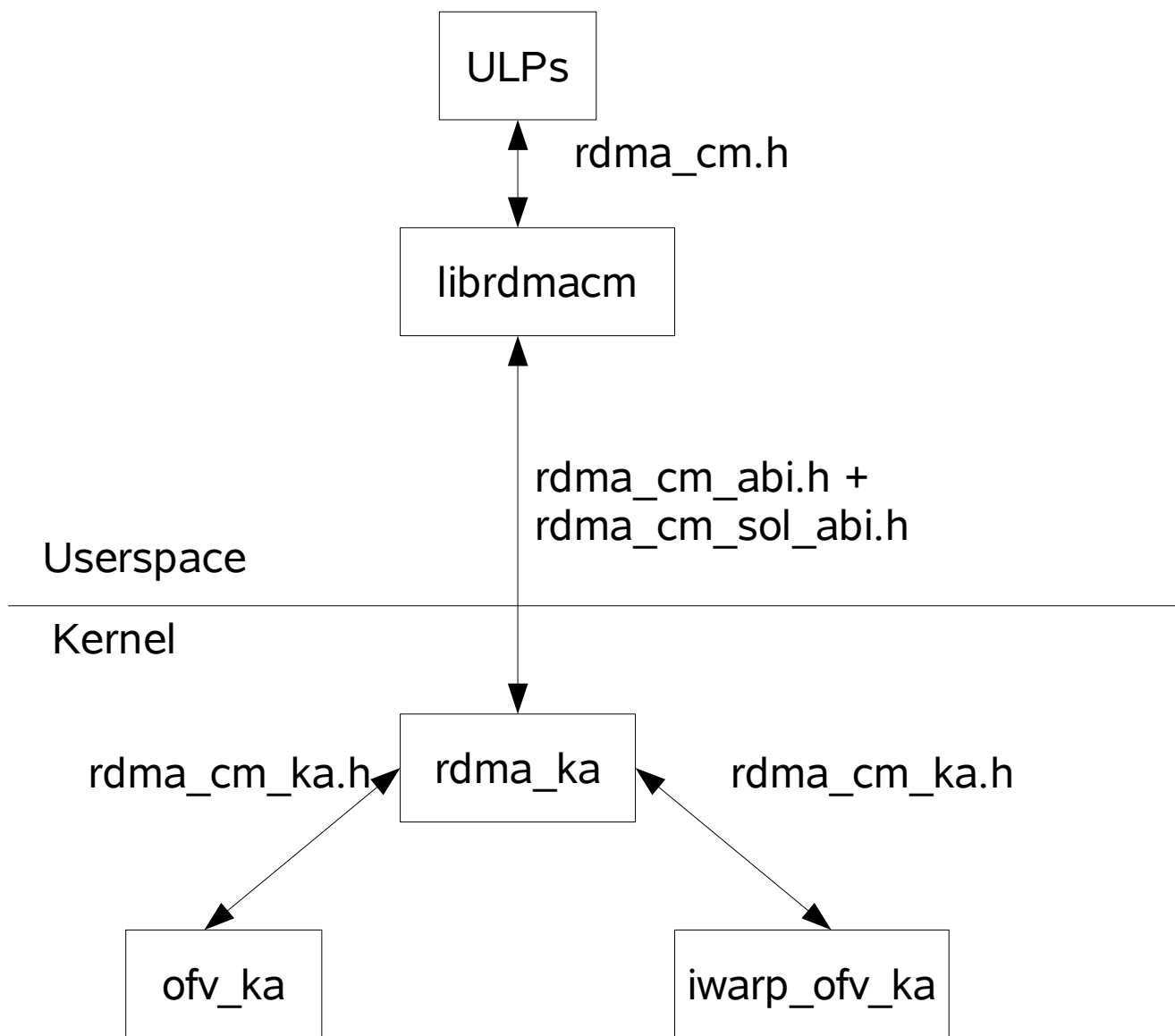
In phase 2 the *ibv\_cmd\_\**() interfaces (defined in *driver.h*) to call back into libibverbs, and the OFED userland/kernel agent interface defined in *kern\_abi.h* and *ib\_user\_verbs.h* are implemented. The Solaris OF-UV HCA plug-in will then call into *libibverbs* via *ibv\_cmd\_\**() interfaces, *libibverbs* will then call into the kernel access module via the interfaces defined in *kern\_abi.h* and *ib\_user\_verbs.h*.

### 7.2 iWARP Support

The iWARP controller plug-in libraries will interface with an iWARP specific kernel Verbs driver for verbs related functionality. The support for iWARP will be based on the architecture for the phase-II of the Solaris OF-UV support.

### 7.3 librdmacm Support

The Solaris *librdmacm* library has to support both IB and iWARP libraries in the second phase of the project. The figure below gives the overall architecture of Solaris *librdmacm* support in phase-II.



The overall architecture of *librdmacm* support is described below :

1. *librdmacm* will provide APIs defined by the OFED *rdma\_cma.h* header.
2. *librdmacm* will interact with a common CM kernel module *rdma\_ka*. The APIs defined in *rdma\_cma\_abi.h* will be used for the interaction.
3. The OF RDMA kernel agent *rdma\_ka*, will provide support for transport independent functionality, like *rdma\_create\_id()*.
4. The OF RDMA kernel agent *rdma\_ka*, will call the OF Verbs kernel access agents for device specific functionality, such as establishing a channel, finding a route, etc. The APIs defined in *rdma\_cm\_ka.h*, will be used for this interaction. The Solaris layered driver interfaces for interacting with the kernel agent modules.
5. The APIs defined in *rdma\_cm\_ka.h* will be the same as *rdma\_cm\_abi.h*, but for the following exceptions :
  - ◆ *rdma\_cm\_ka.h* will not contain transport independent functionality like *CREATE\_ID*
  - ◆ *rdma\_cm\_ka.h* will additionally have an interface, *IS\_IPADDR\_VALID\_FOR\_DEVTYPE*. This interface will check if a local or remote IP address is valid for the device type supported by the kernel agent.
6. The IB OF Verbs kernel agent *ofv\_ka*, will interact with IBTF using the IBTF IP address related APIs. This is expected to be similar for iWARP also.

The interface between the *librdmacm* library and *rdma\_ka* kernel module will also include Solaris specific support. The Solaris specific support extensions should ensure that the IB OF Verbs kernel agent is closed by the *rdma\_cm* kernel agent, when all the IB HCAs have been dynamically unconfigured. Similar support needs to be supported for iWARP also. The details of Solaris specific extensions will be defined in detail at a later stage.

## 7.4 Support for Mellanox Arbel and Hermon Controllers

The Phase-I of the project will support only Mellanox tavor controllers and Arbel controllers in tavor compatibility mode. The support for Mellanox Arbel and Hermon controllers will be based on the Solaris OF-UV phase-II architecture. Userland drivers should be developed for Mellanox Arbel and Hermon controllers based on the phase-II architecture.

## 7.5 Support for future IB HCAs

The phase-II Solaris OF-UV architecture will enable development of third party userland drivers, to support any future HCAs. The interfaces for the userland driver will be similar to the interfaces in the Linux OFED implementation. This will make the porting of userland drivers to Solaris simpler. The section contains details of phase-II of Solaris OF-UV architecture.

## 7.6 SA Access and User MAD library

Support for SA access and User MAD can be provided independent of this project. These libraries can interface with a kernel module which exports the required SA Access and User MAD interfaces to user land. The kernel driver will in turn use the underlying IB Management Framework (IBMF) functionality for MAD transport and SA access. The details of this support is out of scope of the current document.

## 8 Solaris OF-UV test development and testing

The test development for Solaris OF-UV will include the following activities :

1. The OFED release already contains many tests to test the OFED stack. Porting of relevant tests from OFED to Solaris will be required.
2. OFED has API conformance tests for testing the OF-UV API, developed by Mellanox, these will be ported and modified as required.
3. NetPIPE tests will be used for measuring and comparing performance at the OF-UV interface level. The Open MPI/Lustre performance tests will be the main criteria for comparing performance of the Solaris OF-UV implementation.
4. No standard tools to verify inter-operability at the OF-UV interface level. Interoperability will be tested mostly at the Open MPI layer/Lustre.

The project has plans to include :

1. Functional testing
2. Minimal stress testing using 2-4 node clusters.
3. Minimal inter-operability testing.

This testing will be at the OF-UV level, not at the ULP level. ULP testing will be the responsibility of the ULP teams.