Architecture and Usages of Accelio

Eyal Salomon
Mellanox Technologies

2014 OFA Developer Workshop
Sunday, March 30 - Wednesday, April 2, 2014
Monterey CA
What is Accelio in a Nutshell

High-performance, Transport independent, Simple to use Reliable Messaging and RPC Library for Accelerating applications

- Support User space, **Kernel**, C/C++, Java, Python (Future) bindings
- Optimal usage of CPU and Network hardware resources
- Built in fault-tolerance, transaction reliability, and load-balancing
- Integrated into OpenSource (e.g. HDFS, Ceph), and Commercial Storage/DB products in-order to accelerate its transport with minimal development/integration effort
- OpenSource Community project from the ground up:
  - Site: [http://accelio.org](http://accelio.org)
  - Code in: [http://github.com/accelio](http://github.com/accelio)
  - Project/Bug tracking: [http://launchpad.net/accelio](http://launchpad.net/accelio)

March 30 – April 2, 2014  2014 OFA Developer workshop
Accelio Goal

- **Goal**: Provide an easy to use, reliable, scalable, and high performance data/message delivery middleware that maximize efficiency of modern CPU and NIC hardware

- **Key features**:
  - Focus on high-performance asynchronous APIs
  - Reliable message delivery (end to end)
  - Request/Response (Transaction) or Send/Receive models
  - Provide connection and resource abstraction to max scalability and availability
  - Maximize multi-threaded application performance with dedicated HW resources per thread
  - Designed to maximize the benefits of RDMA, hardware offloads, and Multi-core CPUs
  - Will support multiple transport options (RDMA, TCP, ..)
  - Native support for service and storage clustering/scale-out
  - Small message combining
  - Simple and abstract API
Use multiple connections per session:
- maximize CPU core usage/parallelism
- High-availability & Migration
- Scale network bandwidth

Pluggable Transports:
- Code once for multiple HW options
- Seamlessly use RDMA

Abstract, Easy to use API
High Level Transaction Flow

Request (from remote end-point):
* App Header/function, Data in [], Data out []

**Side A (Initiator)**
- Request details + inline data
- Outgoing Data with RDMA or TCP/IP
- Optional Message Arrived Ack
- Returned Data via RDMA or TCP/IP

**Side B (Receiver)**
- Request Notification (with Data)
- Process Request (Async)
- Send Response (Status + returned data)
- Response details + inline data

Next Request ......

* API is asynchronous, multiple requests can be submitted in parallel, and across multiple links & connections
**Accelio Example - Hello Client**

```c
int main(int argc, char *argv[]) {
    struct ...

    /* open one thread context set the polling timeout */
    ctx = xio_context_create(NULL, 0);

    /* create a session and connect to server */
    session = xio_session_create(XIO_SESSION_CLIENT, &attr, url, 0, 0,
                                  &session_data);

    session_data.conn = xio_connect(session, ctx, 0, NULL, &session_data);
    ...
    xio_send_request(session_data.conn, session_data.req);

    /* run the default xio main loop */
    xio_context_run_loop(ctx, XIO_INFINITE);

    /* normal exit phase */
    xio_context_destroy(ctx);

    return 0;
}
```
int on_session_event(struct xio_session *session, struct xio_session_event_data *event_data, void *cb_user_context)
{
    switch (event_data->event) {
    case XIO_SESSION_CONNECTION_TEARDOWN_EVENT:
        xio_connection_destroy(event_data->conn);
        break;
    case XIO_SESSION_TEARDOWN_EVENT:
        xio_session_destroy(session);
        break;
    }
    return 0;
}

int on_response(struct xio_session *session, struct xio_msg *rsp, int more_in_batch, void *cb_prv_data)
{
    struct ...

    process_response(rsp); /* process the incoming message, send a new one */

    xio_release_response(rsp); /* acknowledge xio that response resources can be recycled */
    ...
    xio_send_request(session_data.conn, session_data.req);

    return 0;
}
```c
int main(int argc, char *argv[]) {
    struct ...

    /* create thread context for the server */
    ctx   = xio_context_create(NULL, 0);

    /* bind a listener server to a portal/url */
    server = xio_bind(ctx, &server_ops, url, NULL, 0, &server_data);

    xio_context_run_loop(ctx, XIO_INFINITE);

    /* normal exit phase */
    xio_unbind(server);
    xio_context_destroy(ctx);

    return 0;
}
```
static int on_new_session(struct xio_session *session, struct xio_new_session_req *req, void *cb_prv_data) {
    /* accept new connection request */
    xio_accept(session, NULL, 0, NULL, 0);

    return 0;
}

static int on_new_request(struct xio_session *session, struct xio_msg *req, int more_in_batch, void *cb_prv_data) {
    struct ...

    /* process request and send a response */
    process_request(req);
    /* attach the original request to response and send it */
    response->request = req;

    xio_send_response(response);

    return 0;
}
Accelio Integration With Other Applications/Projects

- Planned
- Under Development

Accelio is adopted as the high-performance, low-latency, Reliable Messaging/RPC library for variety Open-Source and Commercial products, customer projects

- Support multiple bindings (Kernel C, User Space C/C++, Java, Python (future))
Use case 1: XNBD

AcCelio based network block device

- Multi-queue implementation in the block layer for high performance
- Utilizes AcCelio’s facilities and features:
  - Hardware acceleration for RDMA
  - Zero data copy
  - Lockless design
  - Optimal CPU usage
  - Reliable message delivery
- IO operation translation to libaio submit operations to remote device.
- OpenSource Community project from the ground up:
  - Code in: http://github.com/accelio/xnbd
- Prerequisites:
  - AcCelio 1.1 version and above.
  - Kernel version 3.13.1 and above.
Use case 2: R-AIO Remote File Access Application Example

- Provide access to a remote file system by redirecting libaio (async file IO) commands to a remote server (which will issue the IO and return the results to the client)
- Deliver extraordinary performance to remote ram file (/dev/ram)
  - Using 4 CPUs & HW QPs for parallelism
  - Similar performance to local ram file access (i.e. minimal degradation due to communication)

Performance

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Max IOPs</td>
<td>2.5M</td>
</tr>
<tr>
<td>IO Latency</td>
<td>5us</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>6GB/s</td>
</tr>
</tbody>
</table>

https://github.com/accelio/accelio/tree/master/examples/usr/raio
Use case 3: JXIO

JXIO provides the first RDMA API in JAVA

- JXIO is a Java wrapper of Accelio library
- Open source project: https://github.com/accelio/JXIO
- Preserves Accelio’s zero copy and performance all the way
- Every C struct in Accelio is represented by a matching Java class
- Provides 1.5M transactions per second (in Java)
- Reliable message delivery
- Low memory footprint
- Essential component in Mellanox’s HDFS RDMA acceleration solution
Test Configuration

- **Server**
  - HP ProLiant DL380p Gen8
  - 2 x Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
  - 64 GB Memory

- **Adapters**
  - ConnectX3-Pro VPI (IB FDR or 40GbE)
  - ConnectIB 16x PCIe
  - OFED 2.1

- **OS**
  - RedHat EL 6.4
  - Kernel: 2.6.32-358.el6.x86_64

- **Test**
  - Accelio I/O test utility in C, User space
  - Request/Response transactions (RPC)
  - Over 1 or 2 ports, using auto load balancing based on threads
Bandwidth Results

1 I/O Thread
(use single port)

~ 12GB/s with Connect-IB

2 I/O Threads
Transaction Per Second (IOP/s)

Results

March 30 – April 2, 2014
#OFADevWorkshop
Round Trip Latency (Request & Response)

Results

March 30 – April 2, 2014 #OFADevWorkshop
Latency Under Maximum Load (Millions of Messages/Sec)

- **30us @ 1.8M Messages/Sec**
  - 1 I/O Thread
  - (use single port)
  - Hit Max Bandwidth
  - (Link become congested)

- **44us @ ~9M Messages/Sec**
  - 8 I/O Threads
  - With Connect-IB
Open source project

- Initiated by Mellanox
- Partnership
- Companies and Individuals are welcome to join the project and contribute

- Web site: [http://accelio.org](http://accelio.org)
- Code in: [http://github.com/accelio](http://github.com/accelio)
- Project/Bug tracking: [http://launchpad.net/accelio](http://launchpad.net/accelio)
- Email: info@accelio.org
- License: Dual BSD/GPLv2
Thank You