Exploring Custom InfiniBand Drivers for Specialized OS Kernels

Brian Richard Tauro
Kyle C Hale

HExSA

ILLINOIS INSTITUTE OF TECHNOLOGY

HExSA Lab @ IIT (PI Kyle Hale)

- Generally high-performance experimental system software
- Exploration of parallel languages (Julia)
- Specialized operating systems (Nautilus)
- High-performance virtualization (Hybrid virtual machines, Multiverse)

Outline

- **Overview**
- Background
- Prototype Driver
- Experiments
- Summary
- Wishlist

Overview

- We are building a custom InfiniBand device driver for specialized OSes (currently Nautilus)
- …to support specialized use cases
- expose a tailored (app-specific) interface to developers

Why Nautilus?

- Specialized OS kernel framework designed for parallel runtimes (think Unikernel, but runtime-specific, not app-specific)
- Can run alongside Linux by partitioning virtual or physical hardware (**HVM + Multiverse)**
- Good framework for exploring custom drivers (relatively simple codebase)

[A Case for Transforming Parallel Runtimes into OS Kernels. K. Hale, P. Dinda, HPDC ‘15]

[Multiverse (Easy Conversion of Runtime Systems) into OS Kernels via Automatic Hybridization K.Hale, C.Hetland, P.Dinda, ICAC ’17]

Motivation

Why build a custom device driver for InfiniBand ?

Current Diver

- Not Intended for specialized use cases [**General-purpose**]
- Lacks App-specifc Abstractions [**Low-level Interface**]
- Requires additional software stack [**Key Management**]

Not a lot of work in specialized InfiniBand device drivers

Advantages of Custom Device Driver

Key Value Store (KVS)

- RDMA one sided and two sided operations
- Easier to build KVS with high level abstraction [**Get, PUT**]
- Potentially better performance with specialized device drivers

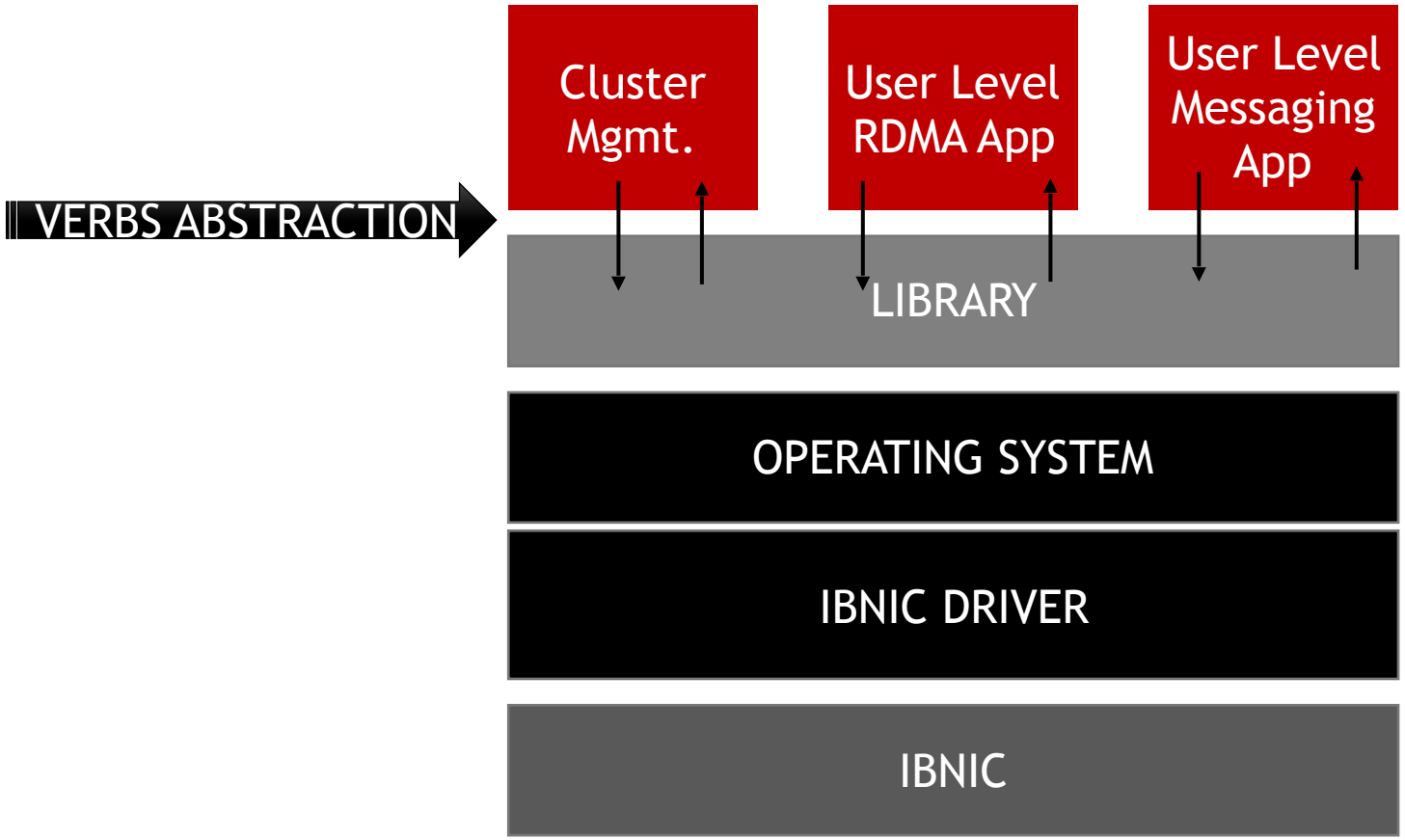
Outline

- Motivation
- **Background**
- Prototype Driver
- Experiments
- Summary
- Wishlist

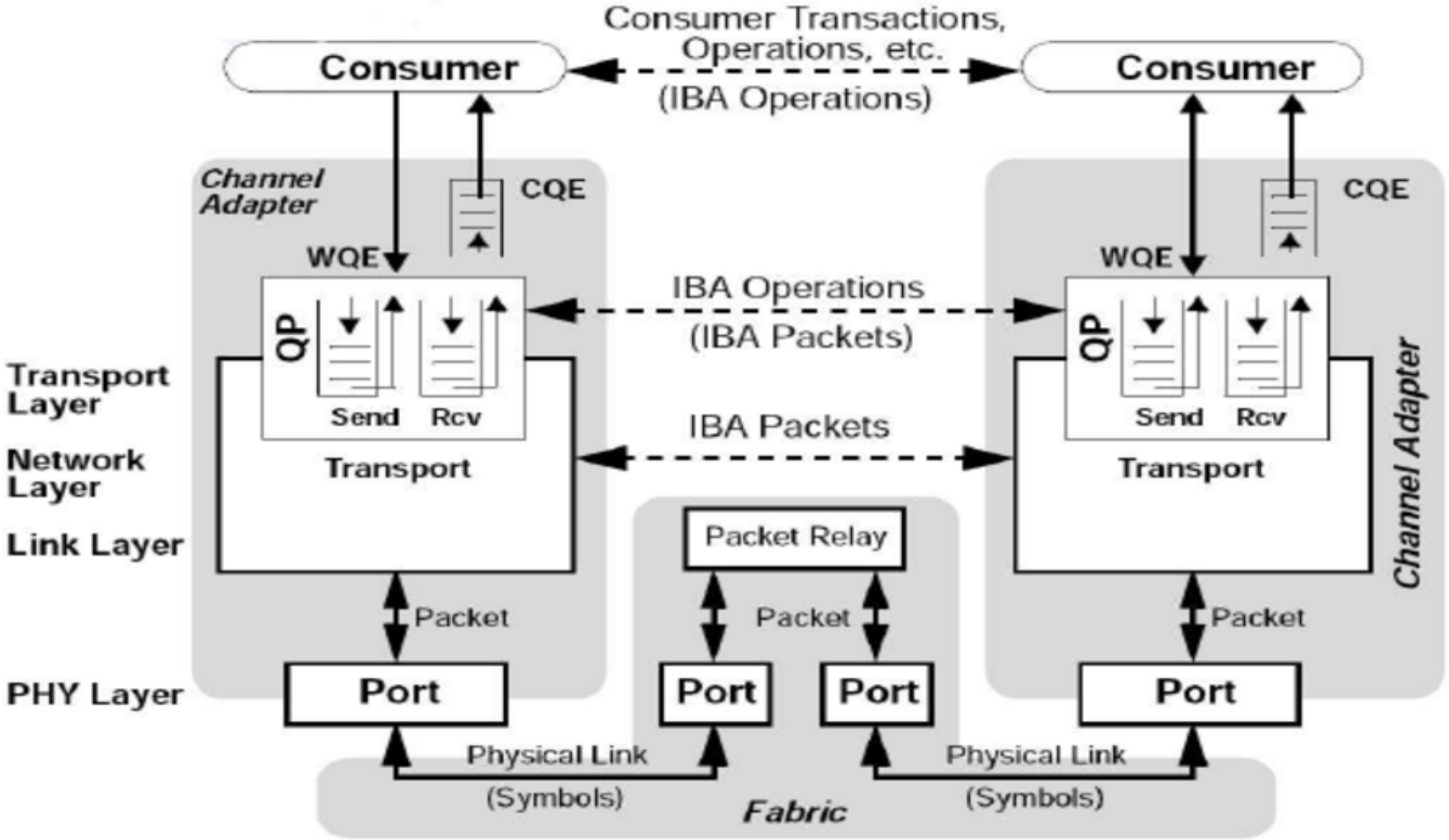
InfiniBand

- ✓ High throughput
- ✓ Extremely low latency
- ✓ Gained huge popularity in data centers and HPC community (RDMA)

Traditional Architecture for InfiniBand



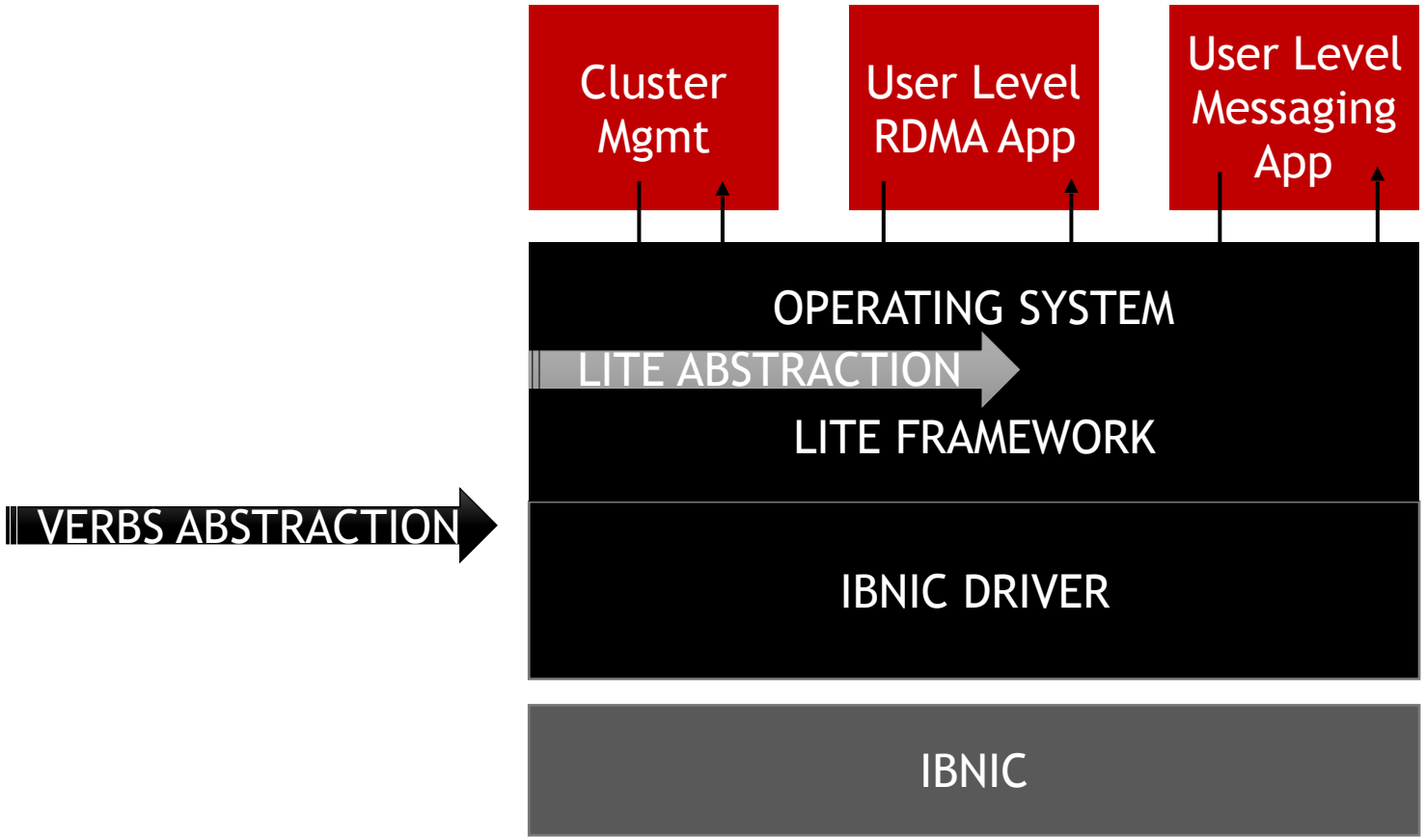
Verbs


Consumer Transactions,
Operations, etc.
(IBA Operations)
Consumer
Consumer
Channel Adapter
CQE
WQE
QP
Send
Rcv
Transport
IBA Operations
(IBA Packets)
IBA Packets
Transport Layer
Network Layer
Link Layer
PHY Layer
Packet
Packet Relay
Port
Physical Link
(Symbols)
Fabric
Channel Adapter

Problems with InfiniBand

- **General Purpose Drivers[abstractions don't match all applications]**
- Exposes a low level interface **[hard to use]**
- Hard to share resources between processes
- Users need to manage keys for memory regions for RDMA **[security concern]**
- Additional software stack [**Management of keys**]

Lite Architecture for InfiniBand



[LITE Kernel RDMA Support for Datacentre Applications , S.Tsai & Y. Zhang, SOSP '17]

What did LITE Achieve?

- Similar performance [**Compared to Directly using verbs**]
- Better Resource Sharing
- Higher Level Abstraction [Management of keys]
- Suitable for both HPC and Data Centric workloads

Outline

- Motivation
- Background
- **Prototype Driver**
- Experiments
- Summary
- Wishlist

Our Goal

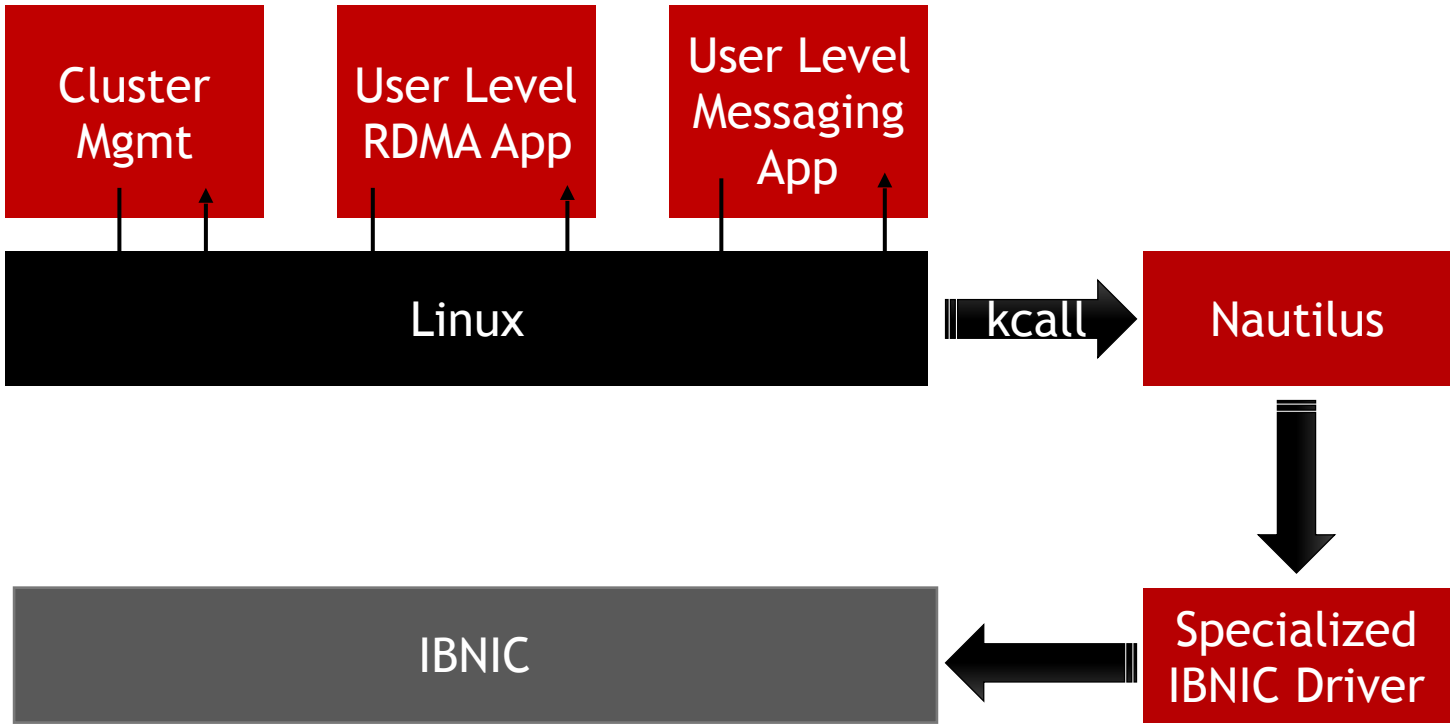
Build a specialized InfiniBand device driver for Nautilus which

Extracts maximum hardware performance

Provides a high-level interface for specialized use cases

Run seamlessly alongside Linux

Hybrid Architecture for InfiniBand

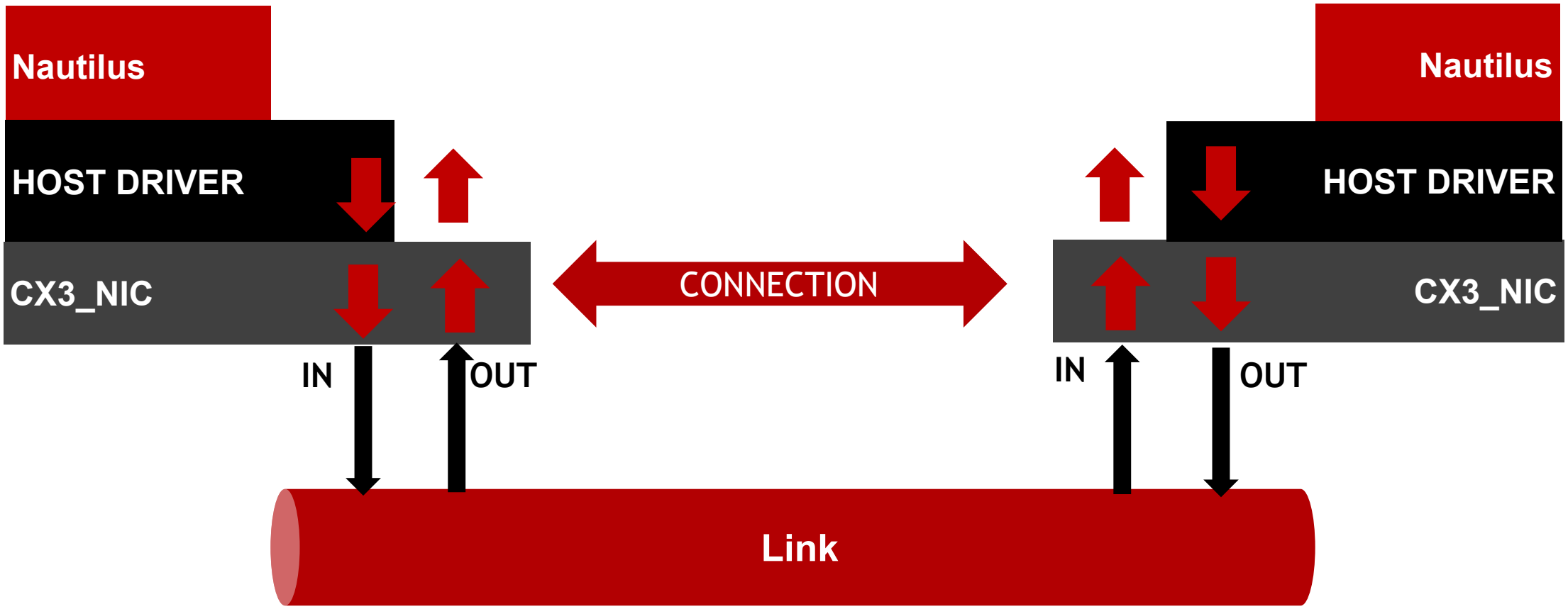


Multiverse (Easy Conversion of Runtime Systems) into OS Kernels via Automatic Hybridization
K.Hale, C.Hetland, P.Dinda, ICAC '17

Prototype

- Our current driver for Nautilus is built with minimal features: bare minimum to send and receive packets
- Current goal: reach the latency limit of the card (disposing of Linux-like abstractions)

So far..


Nautilus
HOST DRIVER
CX3_NIC
IN
OUT
CONNECTION
Nautilus
HOST DRIVER
CX3_NIC
IN
OUT
Link

Why Chameleon?

✓ Chameleon is one of the few cloud providers which provide access to InfiniBand Cards

✓ InfiniBand hardware is expensive

✓ Excellent Platform for research

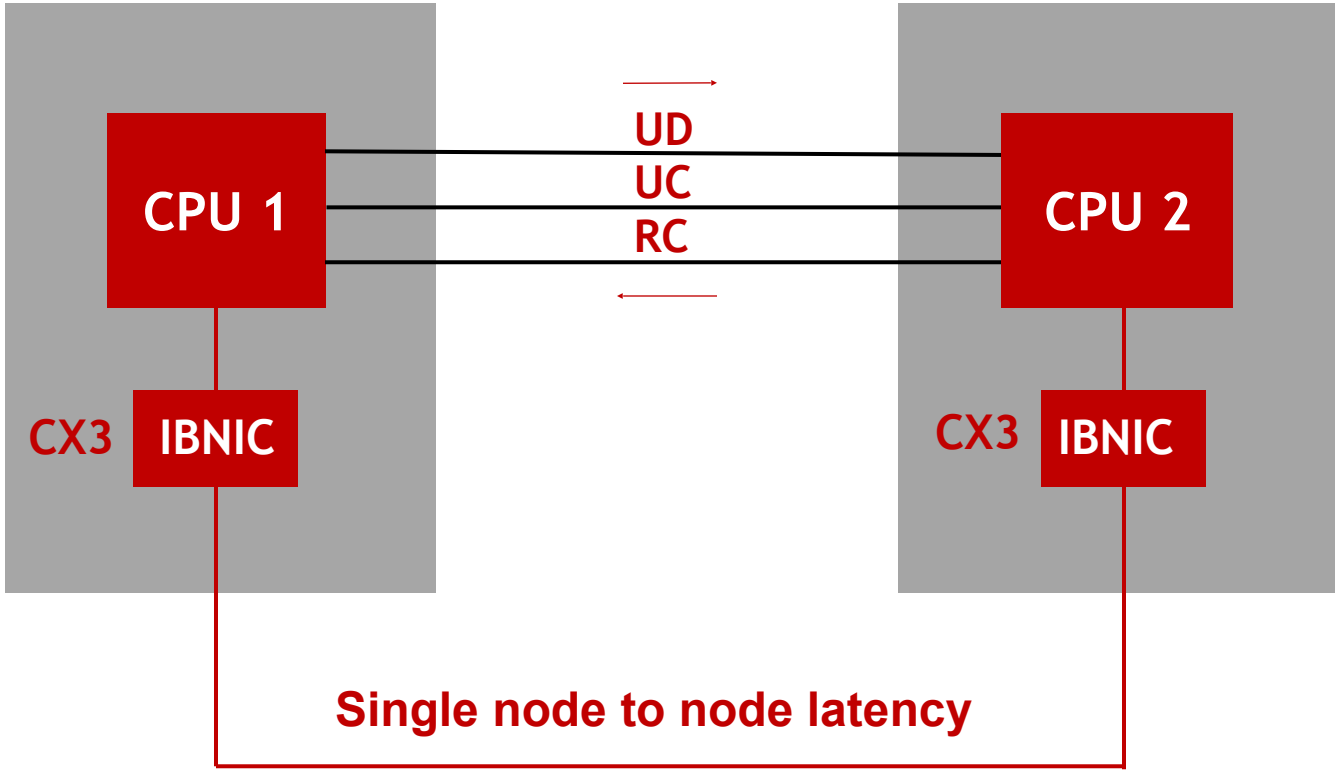
✓ Strong community support

✓ Reliable

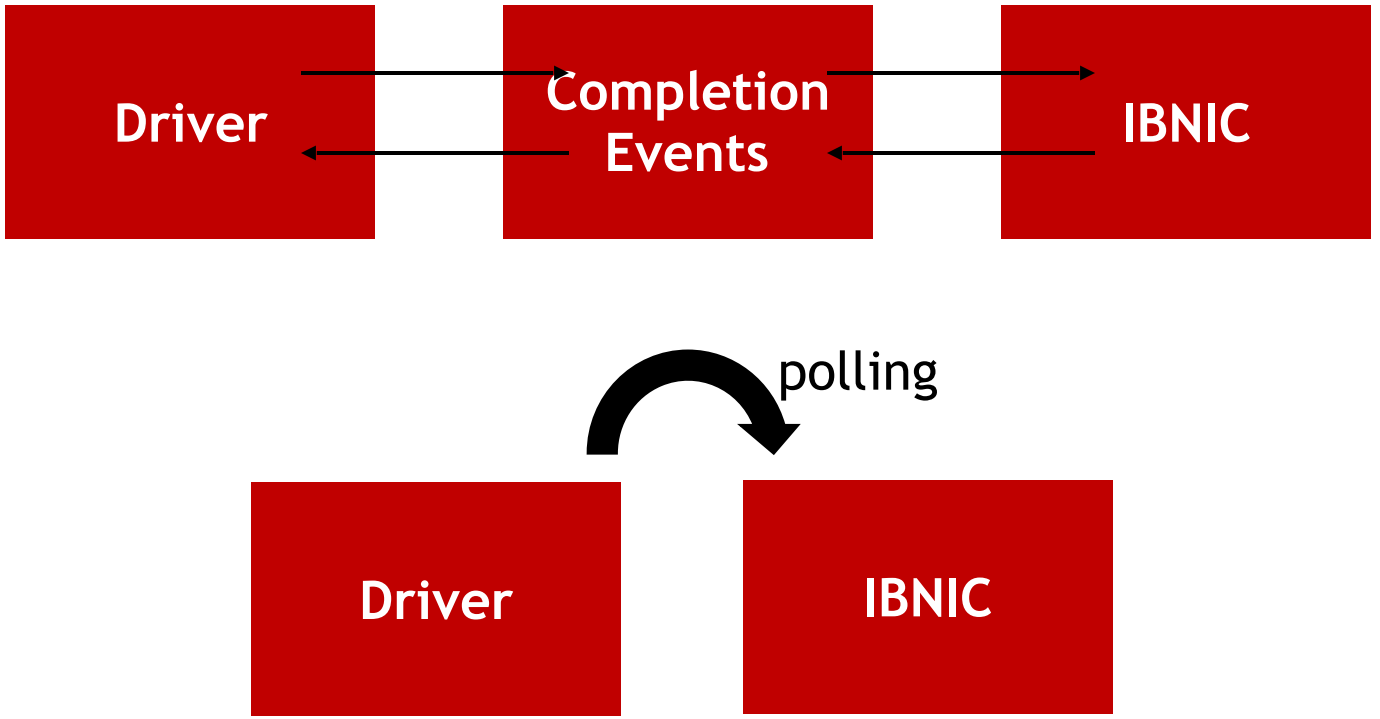
Outline

- Motivation
- Background
- Prototype Driver
- **Experiments**
- Summary
- Wishlist

Experiments (Latency)



Experiments (Events vs Polling)



Event driven interrupts vs Polling in Nautilus

Setup

Approach :

- Run Nautilus directly on **Compute Haswell IB Connect x3** nodes provided by Chameleon.
- Nautilus does not have ssh and user space **>** print on serial console
- only way to see serial output is on the console web interface **>** web interface does not always work
- Web interface is slow **>** difficult to script it

Alternative :

Run the experiments in a KVM virtual machine (on top of Cham. bare-metal instance) with passthrough access to the IB card

Results

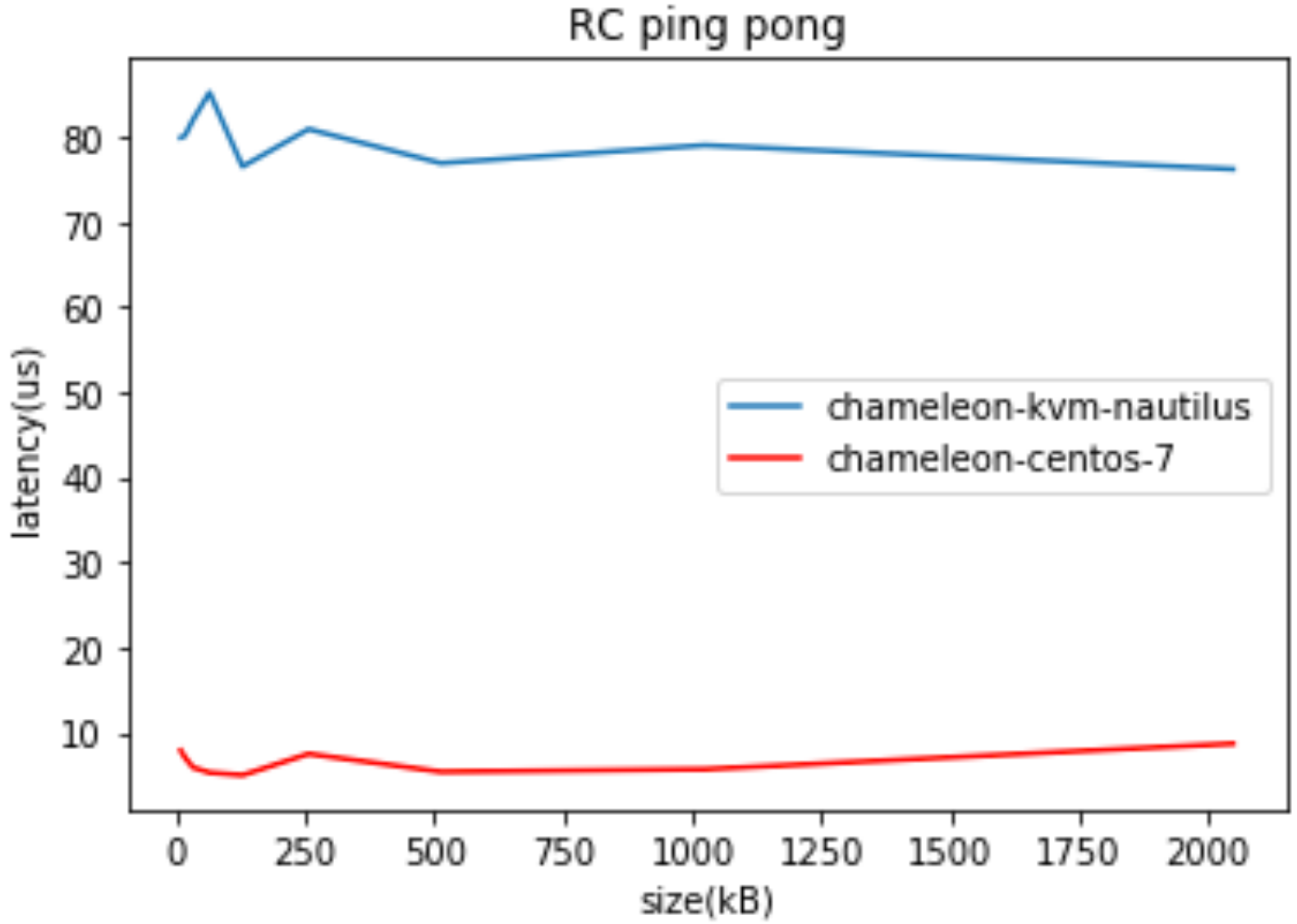
- RTT over KVM in Chameleon for Nautilus takes about **40 - 60 μsec.**

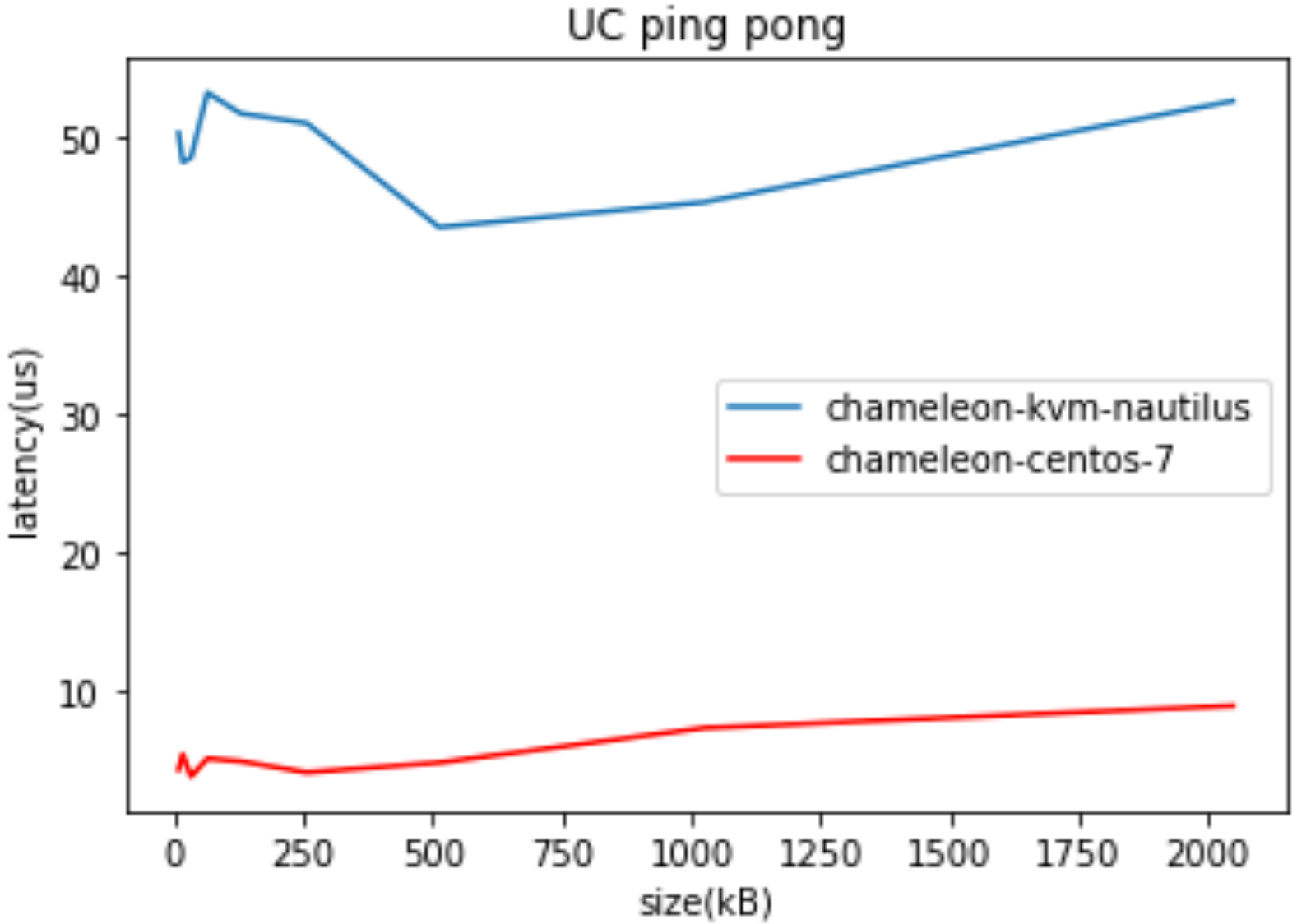
  ➡ could be due to the interrupt overhead caused by KVM (guess).

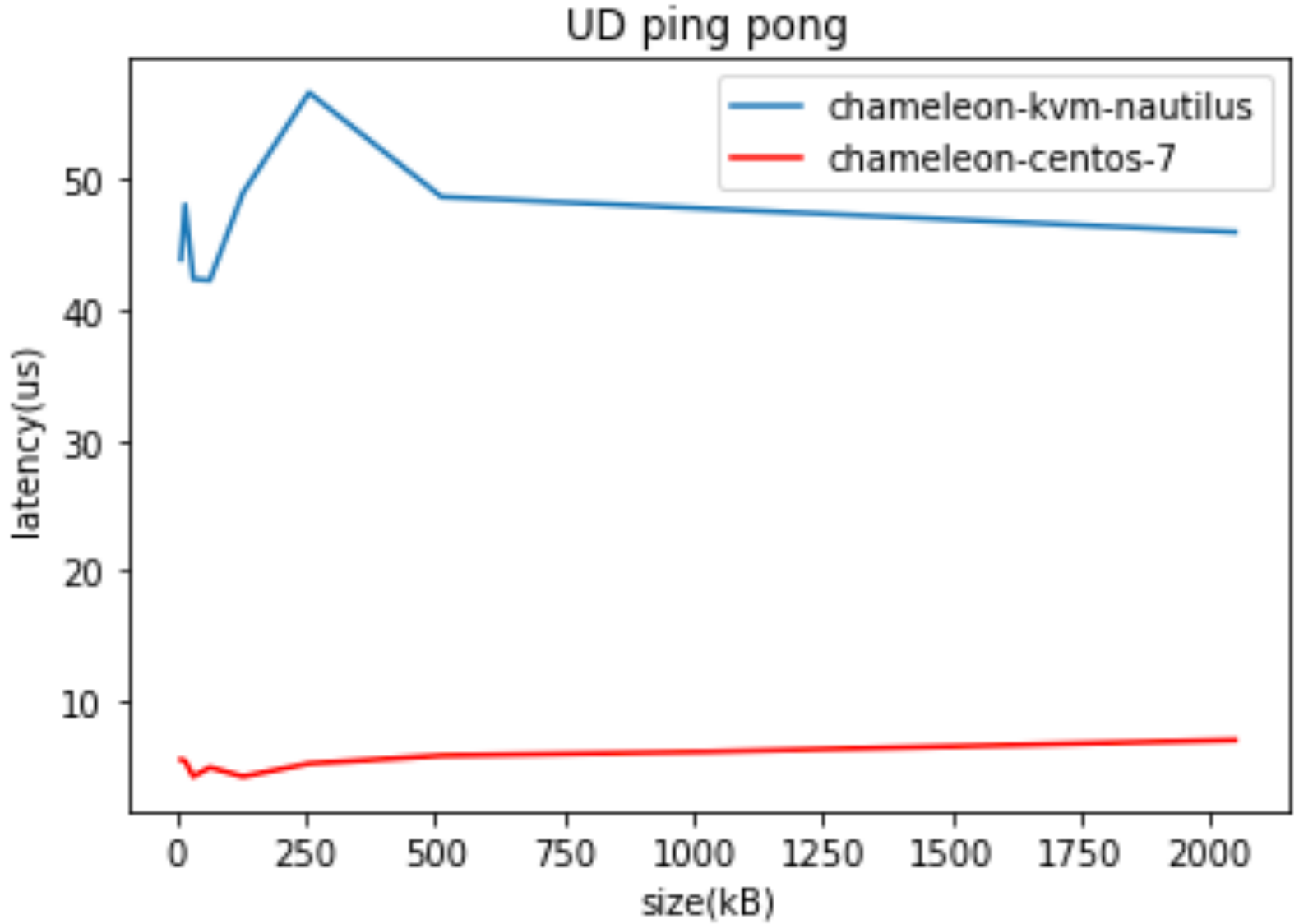
- We also now have our own physical setup @ IIT: two machines with ConnectX-3 cards connected directly via an unmanaged IB switch

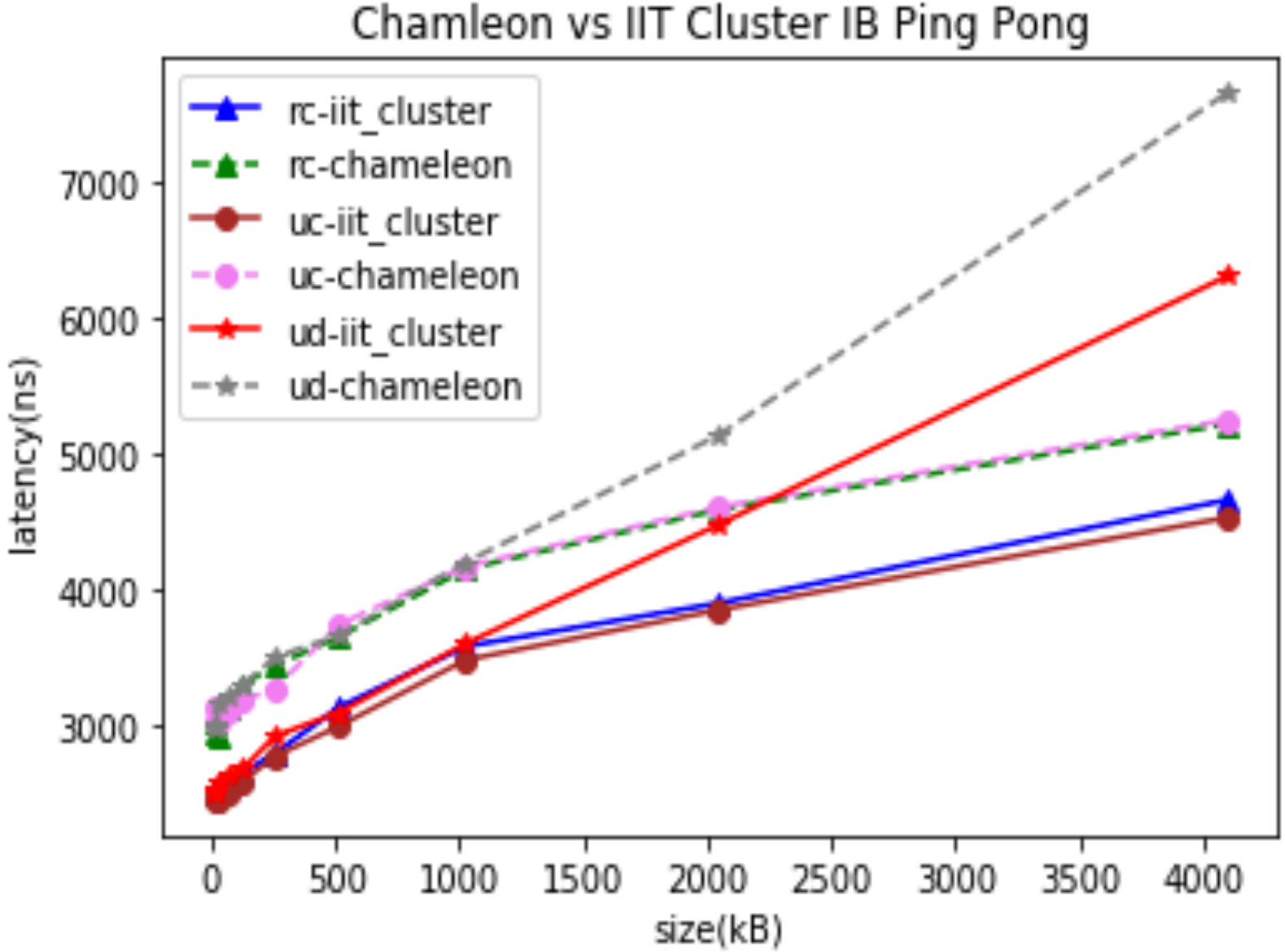
Transport Types

RC	UC	UD
Reliable Channel	Unreliable Channel	Unreliable Datagram
End to End Connection needed between QP	No Ack on Packet delivery but still end to end connection required	Unreliable Datagram Similar to UDP
RDMA	RDMA Write	No RDMA


RC ping pong
latency(us)
80
70
60
50
40
30
20
10
0
250
500
750
1000
1250
1500
1750
2000
size(kB)
chameleon-kvm-nautilus
chameleon-centos-7


UC ping pong
latency(us)
50
40
30
20
10
0
250
500
750
1000
1250
1500
1750
2000
size(kB)
chameleon-kvm-nautilus
chameleon-centos-7


UD ping pong
chameleon-kvm-nautilus
chameleon-centos-7
latency(us)
50
40
30
20
10
0
250
500
750
1000
1250
1500
1750
2000
size(kB)


Chamleon vs IIT Cluster IB Ping Pong
rc-iit_cluster
rc-chameleon
uc-iit_cluster
uc-chameleon
ud-iit_cluster
ud-chameleon
latency(ns)
7000
6000
5000
4000
3000
0
500
1000
1500
2000
2500
3000
3500
4000
size(kB)

Outline

- Motivation
- Background
- Prototype Driver
- Experiments
- **Summary**
- Wishlist

Challenges

- Running Nautilus bare metal in our cluster (bug hunting)
- Scriptable setup for experiments in nautilus
- Tuning the driver for extremely low latency

Future Work

- Specialized InfiniBand driver tailored for specific work loads
- For example, high-performance, in-memory KVS with custom RDMA support
- Get hybrid (Linux/Nautilus partition) working with driver

Outline

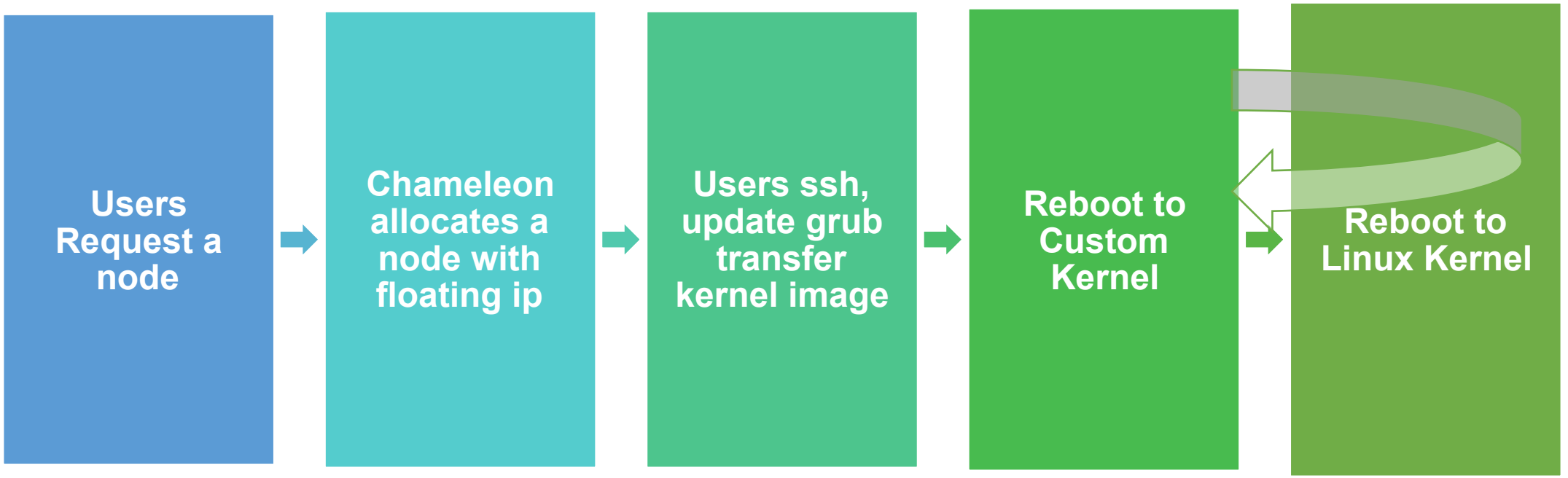
- Motivation
- Background
- Prototype Driver
- Experiments
- Summary
- **Wishlist**

Wish List

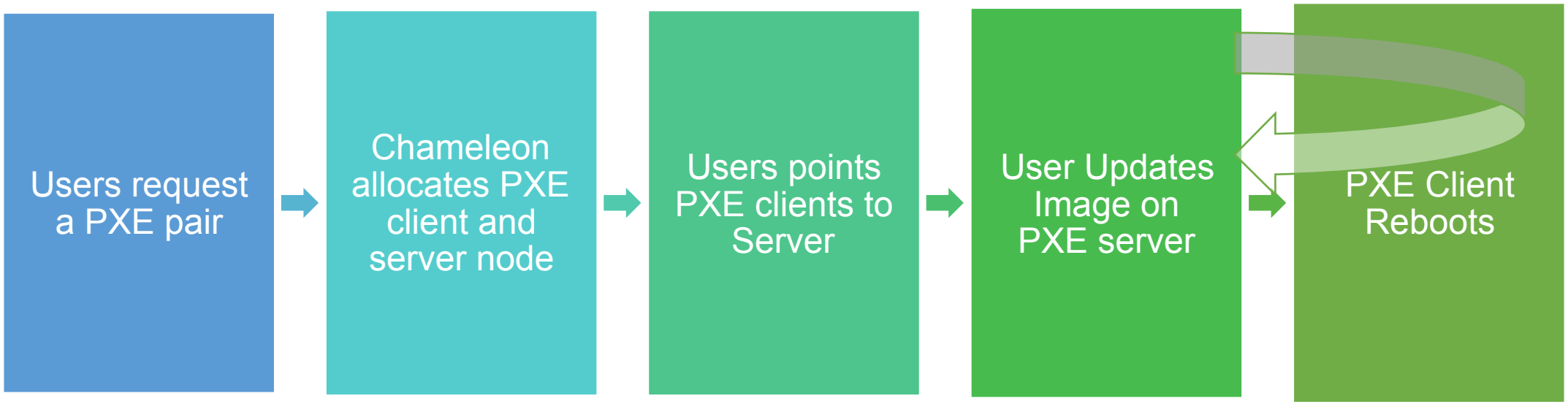
A way to access BMC, or an interface to setup PXE boot for development of specialized kernels.

- Scriptable booting for custom OSes (with serial output)
- Programmatic way to access serial console
- Flexible ways to set up PXE server/client configurations

Current Chameleon Workflow for Custom OS Boot



Proposed PXE Workflow for Custom OS Boot



THANKS!

me: btauro@hawk.iit.edu
kyle: khale@cs.iit.edu

our lab: http://hexsa.halek.co

Nautilus codebase: https://github.com/hexsa-lab/nautilus

HExSA

ILLINOIS INSTITUTE OF TECHNOLOGY