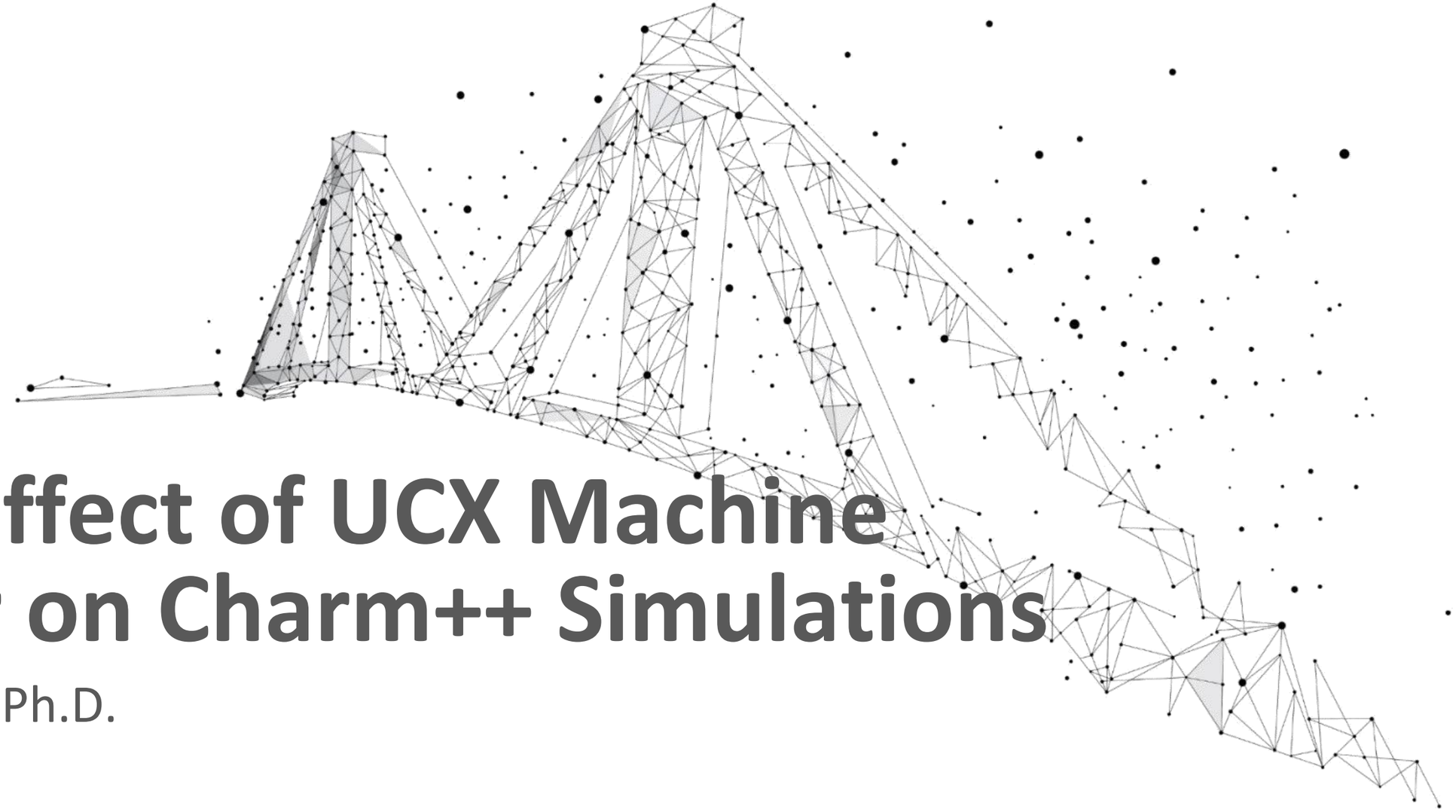


The Effect of UCX Machine Layer on Charm++ Simulations

Yong Qin, Ph.D.

May 2019

Agenda

- Introduction to UCX
- UCX Machine Layer
- Performance Evaluations
- Summary and Conclusions

UCX

UCX Introduction

- Unified Communication X (UCX) is an open-source communication framework
- UCX is a communication middleware with rich and comprehensive API
- UCX is a production grade communication framework for data centric and high-performance applications
- UCX is a co-design enabler and collaboration between industry, laboratories, and academia

WEB:
www.openucx.org
https://github.com/openucx/ucx
Mailing List:
https://elist.ornl.gov/mailman/listinfo/ucx-group
ucx-group@elist.ornl.gov

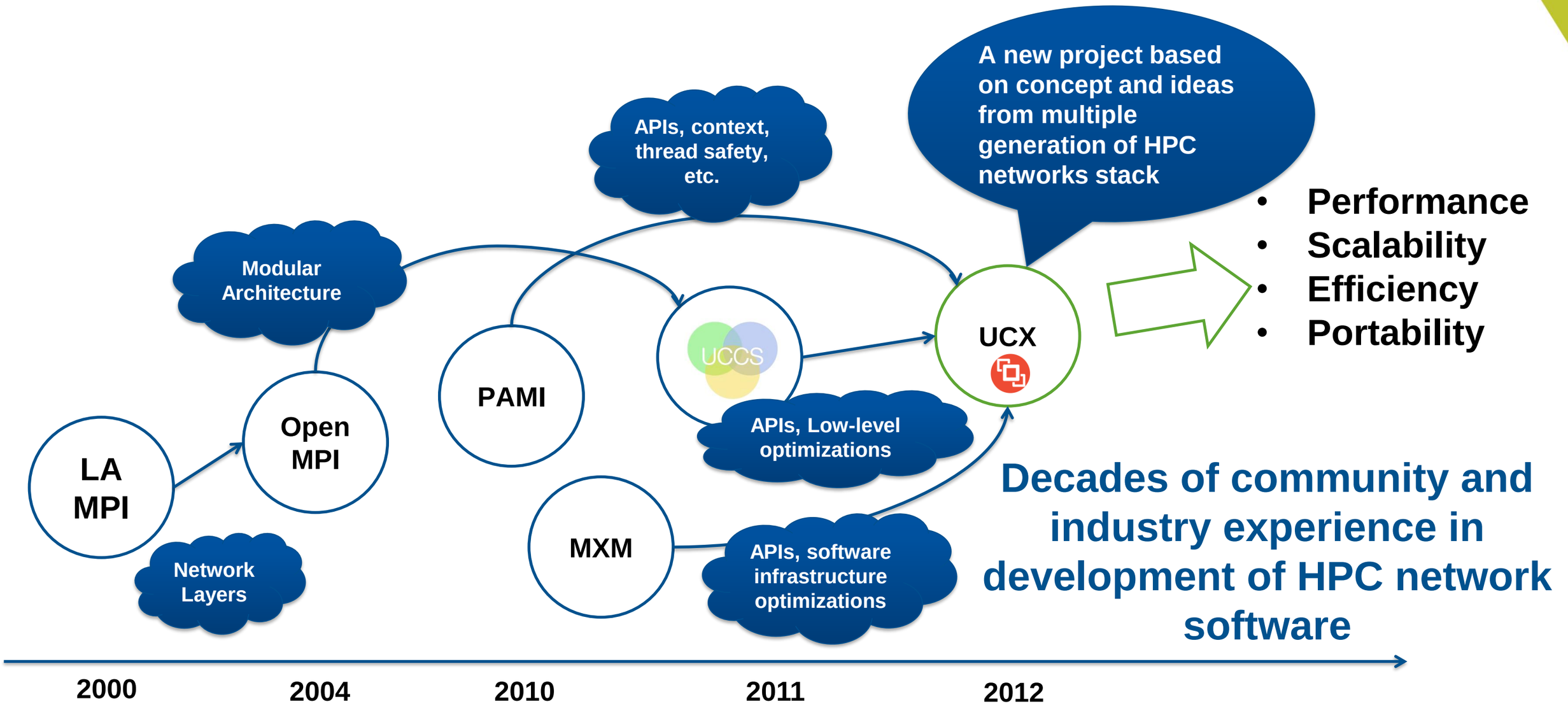
UCF Consortium

- Mission:
  - Collaboration between industry, laboratories, and academia to create production grade communication frameworks and open standards for data centric and high-performance applications

- Projects
  - UCX – Unified Communication X
  - Open RDMA

- Board members
  - Jeff Kuehn, UCF Chairman (Los Alamos National Laboratory)
  - Gilad Shainer, UCF President (Mellanox Technologies)
  - Pavel Shamis, UCF treasurer (ARM)
  - Brad Benton, Board Member (AMD)
  - Duncan Poole, Board Member (Nvidia)
  - Pavan Balaji, Board Member (Argonne National Laboratory)
  - Sameh Sharkawi, Board Member (IBM)
  - Dhabaleswar K. (DK) Panda, Board Member (Ohio State University)
  - Steve Poole, Board Member (Open Source Software Solutions)

UCX - History



- Performance
- Scalability
- Efficiency
- Portability

Decades of community and industry experience in development of HPC network software

UCX Framework Mission

- Collaboration between industry, laboratories, government (DoD, DoE), and academia
- Create open-source production grade communication framework for HPC applications
- Enable the highest performance through co-design of software-hardware interfaces

API

Exposes broad semantics that target data centric and HPC programming models and applications

Performance oriented

Optimization for low-software overheads in communication path allows near native-level performance

Production quality

Developed, maintained, tested, and used by industry and researcher community

Community driven

Collaboration between industry, laboratories, and academia

Research

The framework concepts and ideas are driven by research in academia, laboratories, and industry

Cross platform

Support for Infiniband, Cray, various shared memory (x86-64, Power, ARMv8), GPUs

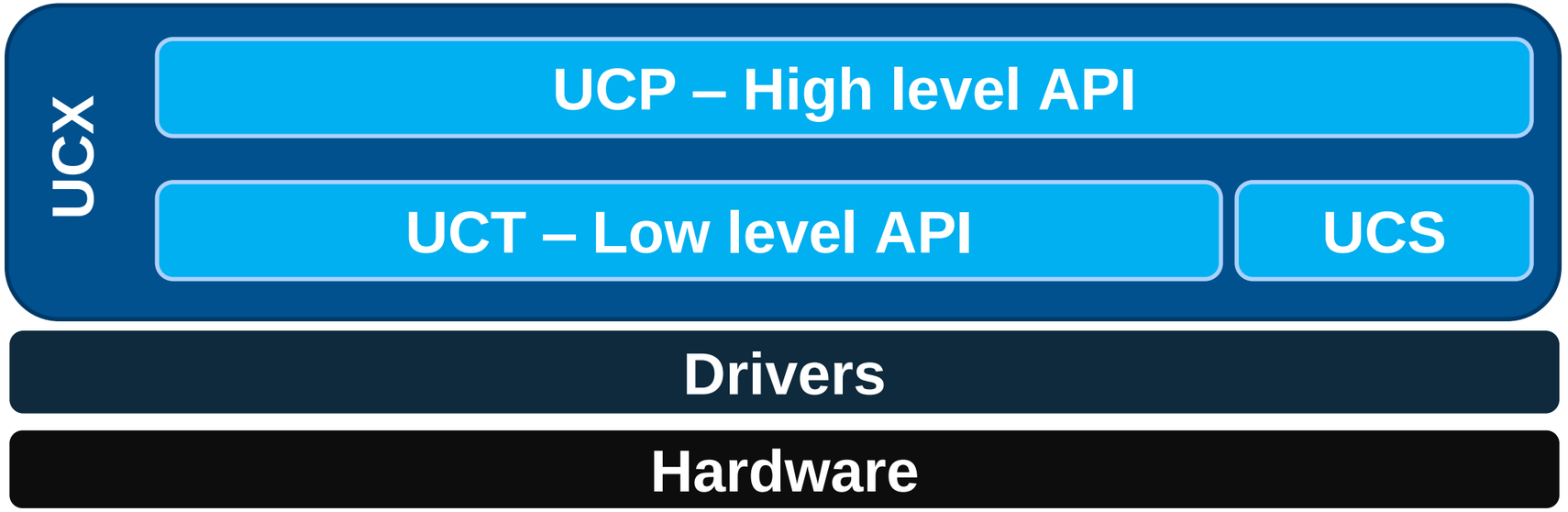
Co-design of Exascale Network APIs

UCX Framework

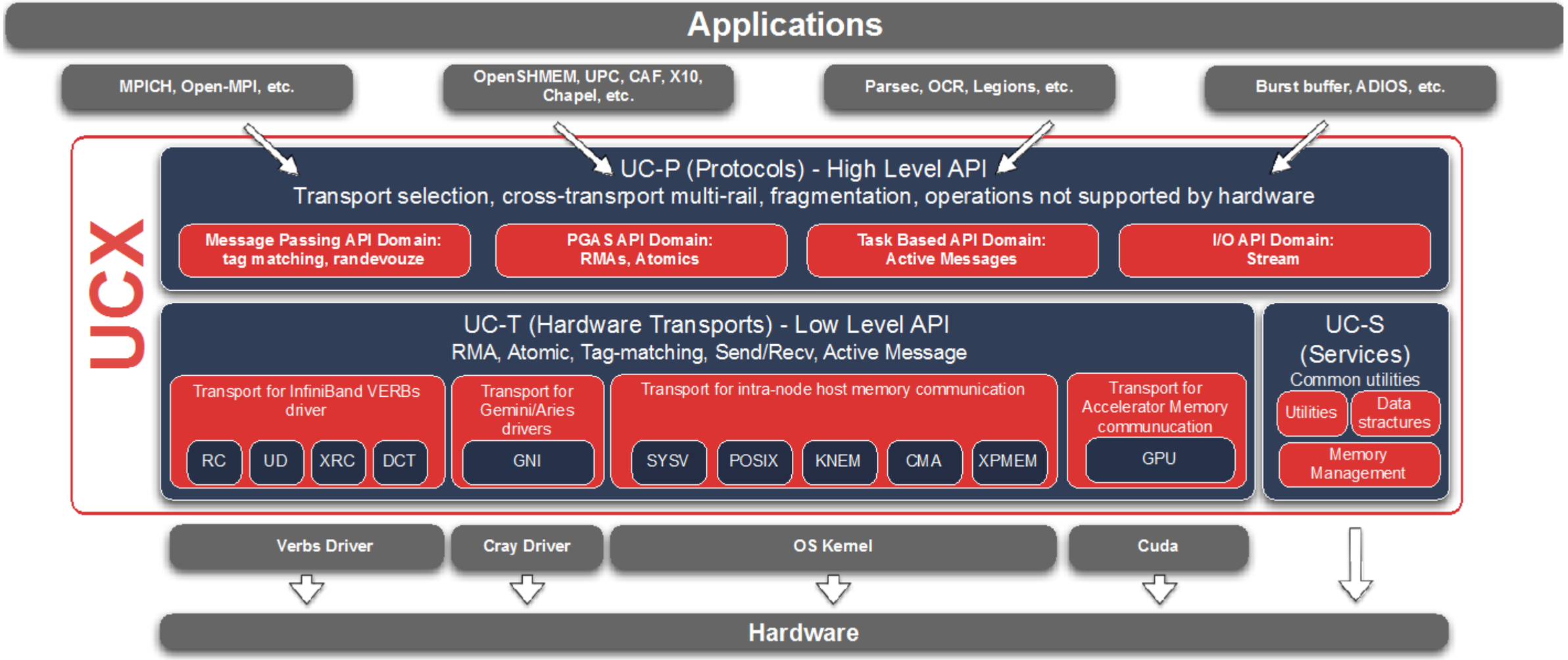
- UCX is a framework for network APIs and stacks
- UCX aims to unify the different network APIs, protocols and implementations into a single framework that is portable, efficient and functional
- UCX doesn’t focus on supporting a single programming model, instead it provides APIs and protocols that can be used to tailor the functionalities of a particular programming model efficiently
- When different programming paradigms and applications use UCX to implement their functionality, it increases their portability. As just implementing a small set of UCX APIs on top of a new hardware ensures that these applications can run seamlessly without having to implement it themselves

UCX Architecture

- **UCX framework** is composed of **three main components**.
- **UCP** layer is the protocol layer and supports all the functionalities exposed by the high-level APIs, meaning it emulates the features that are not implemented in the underlying hardware
- **UCT** layer is the transport layer that aims to provides a very efficient and low overhead access to the hardware resources
- **UCS** is a service layer that provides common data structures, memory management tools and other utilities



UCX High-Level Overview



UCX Transports and Architecture

- Transports
  - InfiniBand/RoCE
    - RC, DC, UD
  - Shared memory
    - Posix, SysV, knem, cma, xpmem
- Architectures
  - x86_64
  - aarch64
  - ppc64le
- Simple and consistent network APIs
  - Tag Matching
  - RMA
  - AMO
  - AM
  - Stream
- Networks
  - InfiniBand,
  - Ethernet (TCP/IP, RoCE),
  - Shared Memory
  - uGNI
- Highly optimized Out of the Box

UCX Machine Layer for Charm++

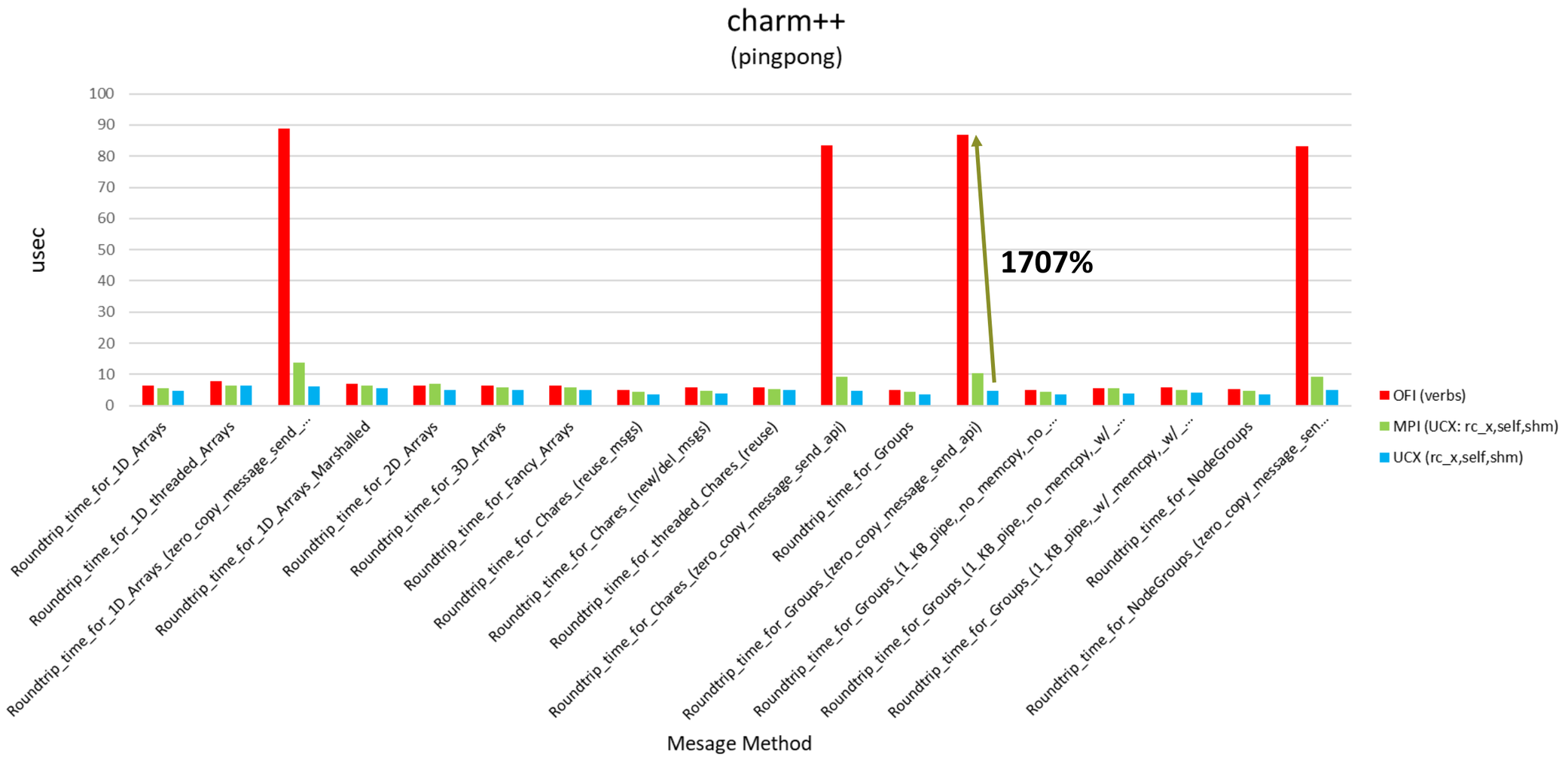
UCX as a Machine Layer

- UCX can be a perfect fit for Charm++ machine layer
- UCX provides ultra low latency and high bandwidth sitting on top of InfiniBand stack
- UCX provides much less intrusive and close-to hardware API for one-sided communications than MPI
- UCX machine layer is implemented as LRTS layer and supports the following modes
  - SMP
  - ONESIDED
  - DIRECT_ONESIDED

Performance Evaluation

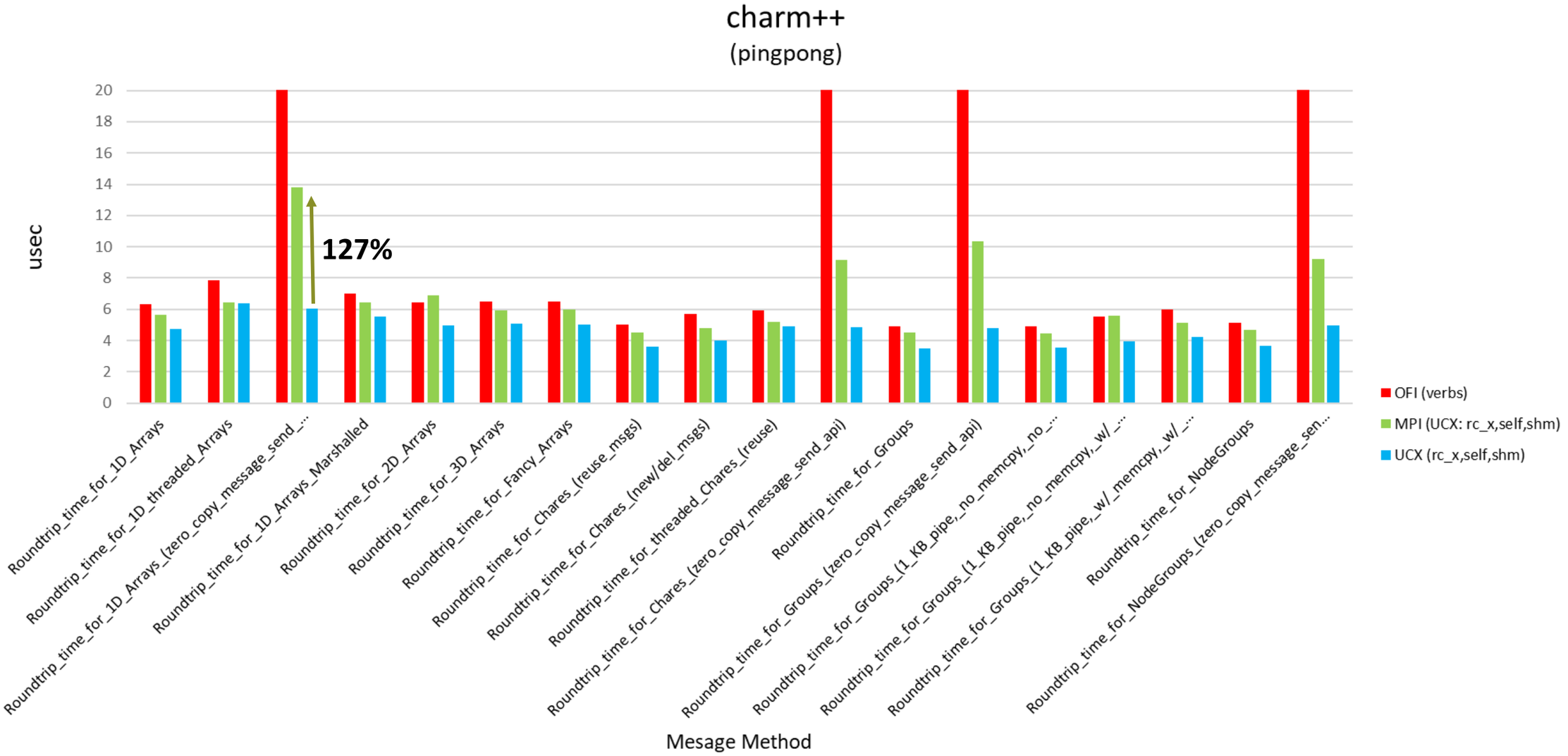
Charm++ PingPong (vs. Verbs)

- UCX machine layer shows major performance improvements over Verbs



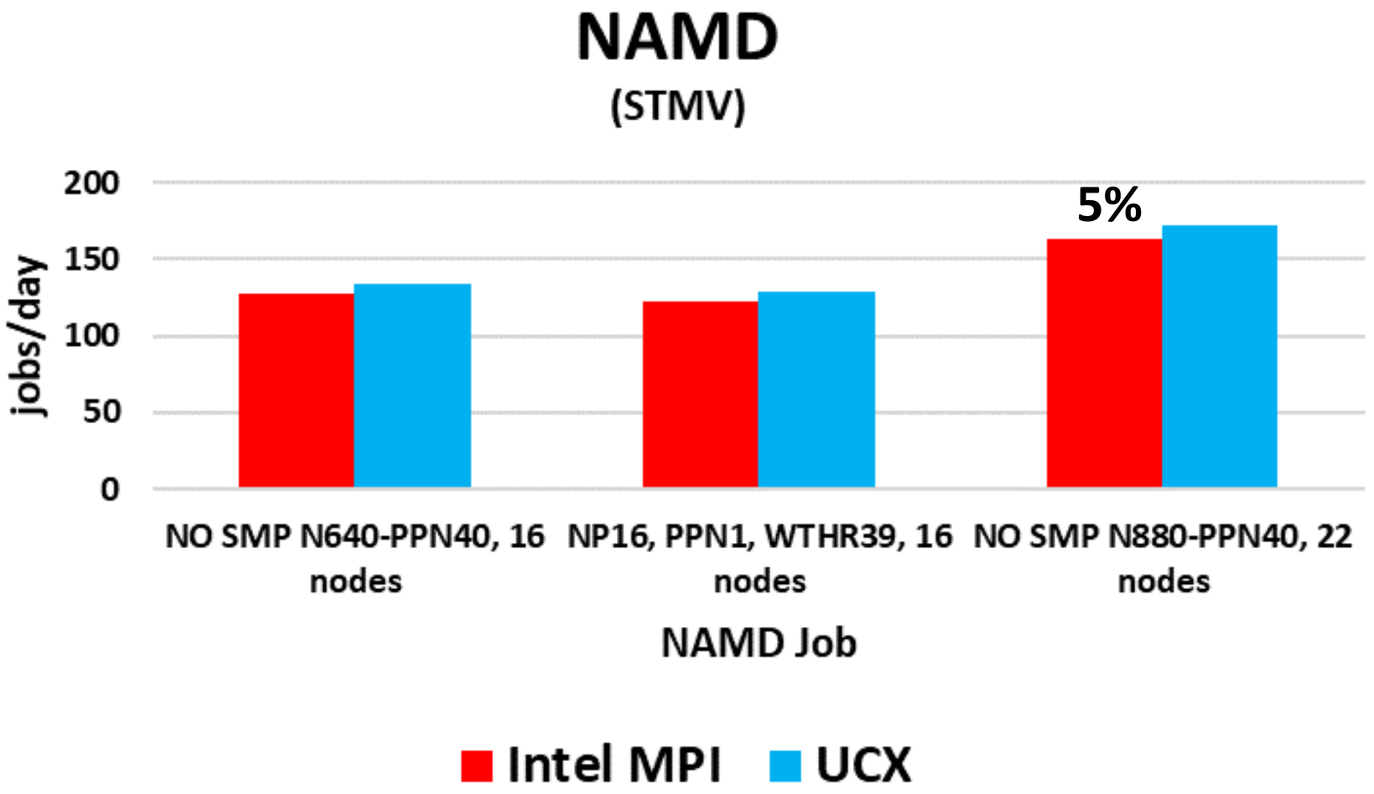
Charm++ PingPong (vs. MPI)

- UCX machine layer shows major performance improvements over MPI



Application workload - NAMD

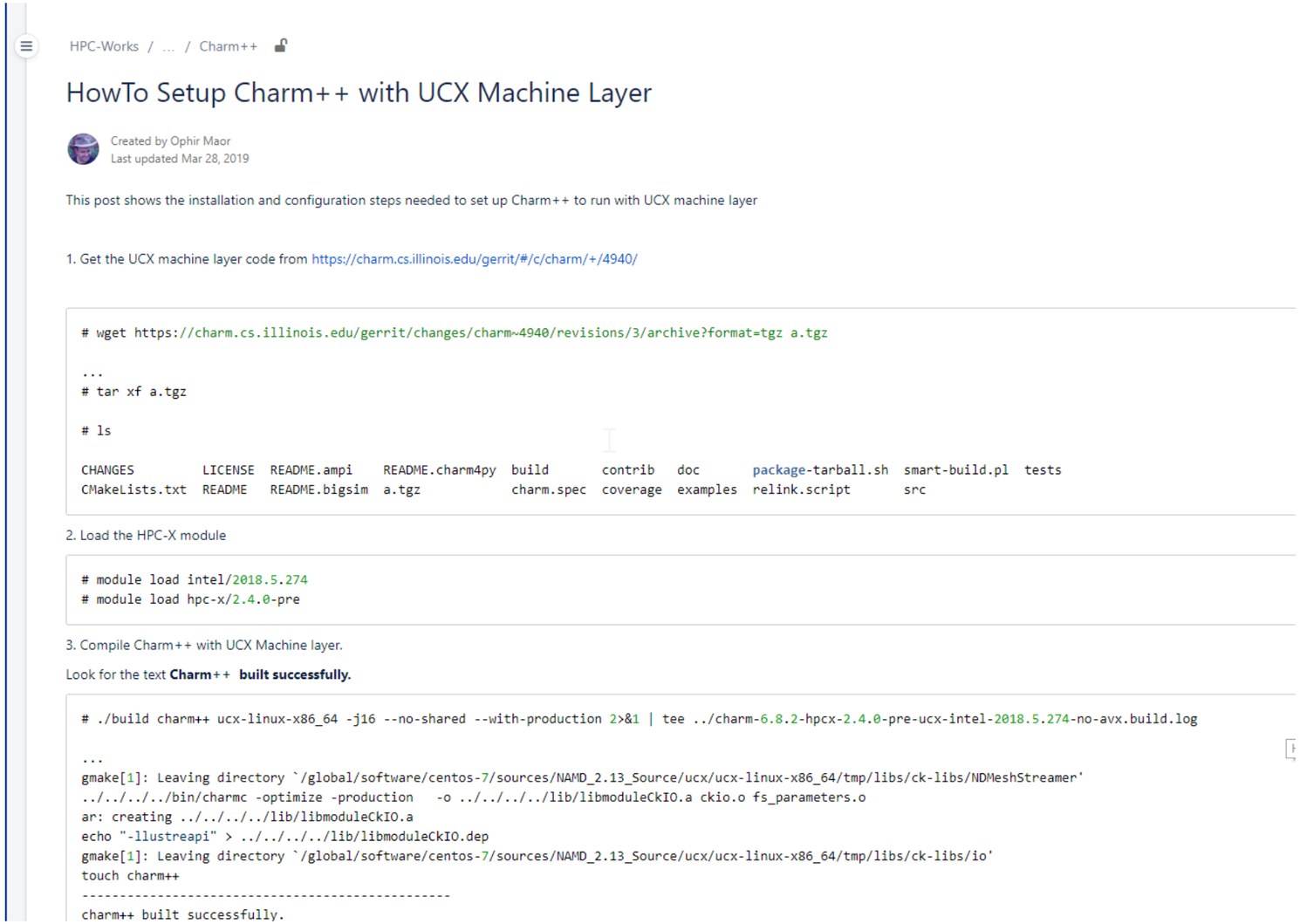
- With STMV[1] NAMD Input:
  - UCX machine layer shows ~5% improvements over Intel MPI



[1] https://www.ks.uiuc.edu/Research/namd/utilities/

How To setup Charm++ with UCX Machine Layer

- Getting started procedure can be found here:
  https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/pages/1040449537/HowTo+Setup+Charm+with+UCX+Machine+Layer



HPC-Works / ... / Charm++

HowTo Setup Charm++ with UCX Machine Layer

Created by Ophir Maor
Last updated Mar 28, 2019

This post shows the installation and configuration steps needed to set up Charm++ to run with UCX machine layer

1. Get the UCX machine layer code from https://charm.cs.illinois.edu/gerrit/#/c/charm/+/4940/

```
# wget https://charm.cs.illinois.edu/gerrit/changes/charm~4940/revisions/3/archive?format=tgz a.tgz

...
# tar xf a.tgz

# ls

CHANGES         LICENSE  README.ampi    README.charm4py  build       contrib   doc       package-tarball.sh  smart-build.pl  tests
CMakeLists.txt  README   README.bigsim  a.tgz            charm.spec  coverage  examples  relink.script       src
```

2. Load the HPC-X module

```
# module load intel/2018.5.274
# module load hpc-x/2.4.0-pre
```

3. Compile Charm++ with UCX Machine layer.

Look for the text **Charm++ built successfully.**

```
# ./build charm++ ucx-linux-x86_64 -j16 --no-shared --with-production 2>&1 | tee ../charm-6.8.2-hpcx-2.4.0-pre-ucx-intel-2018.5.274-no-avx.build.log

...
gmake[1]: Leaving directory `/global/software/centos-7/sources/NAMD_2.13_Source/ucx/ucx-linux-x86_64/tmp/libs/ck-libs/NDMeshStreamer'
../../../../bin/charmc -optimize -production   -o ../../../../lib/libmoduleCkIO.a ckio.o fs_parameters.o
ar: creating ../../../../lib/libmoduleCkIO.a
echo "-llustreapi" > ../../../../lib/libmoduleCkIO.dep
gmake[1]: Leaving directory `/global/software/centos-7/sources/NAMD_2.13_Source/ucx/ucx-linux-x86_64/tmp/libs/ck-libs/io'
touch charm++
--------------------------------------------------
charm++ built successfully.
```

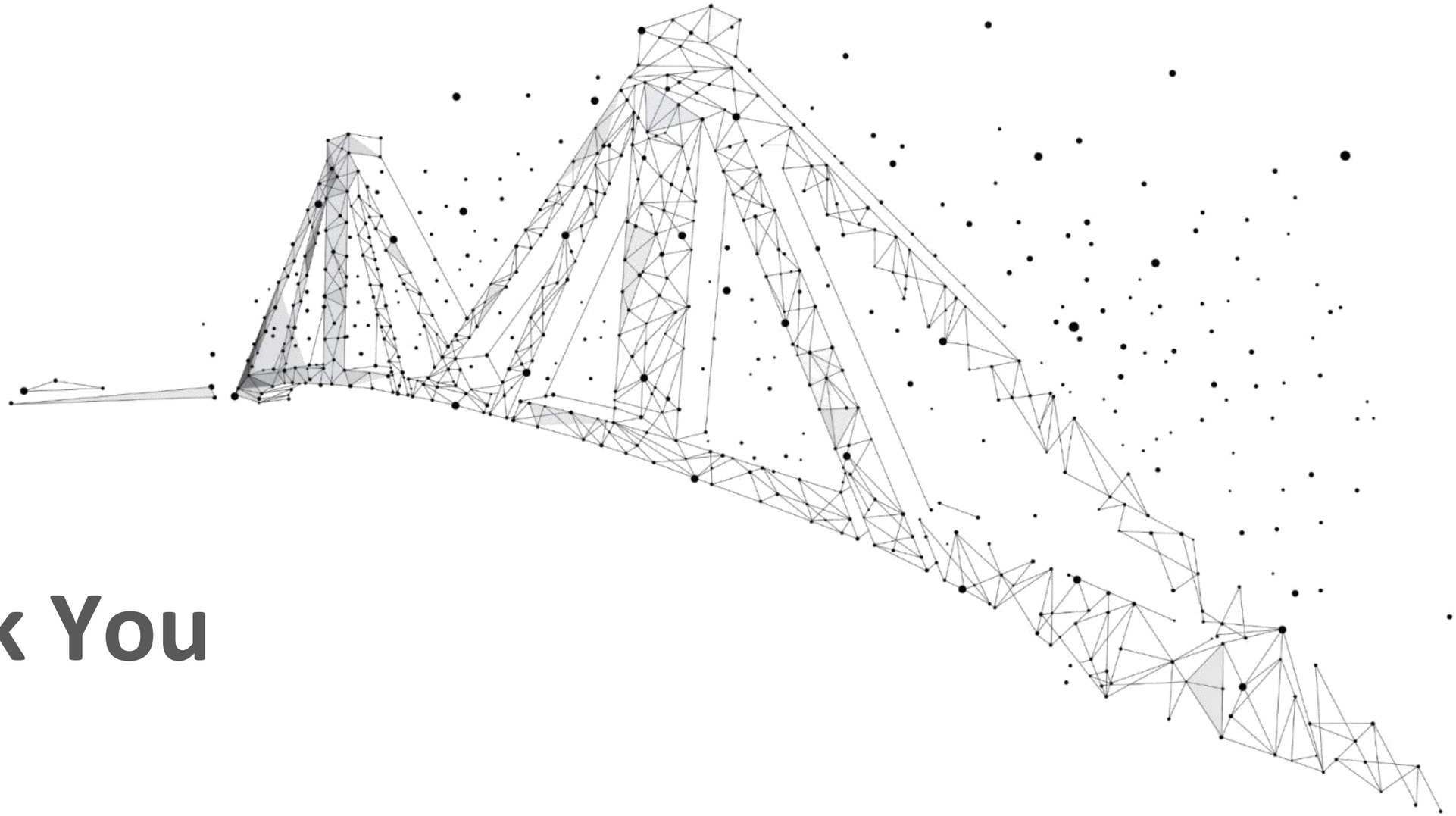
Summary and Conclusions

Summary and Conclusions

- Microbenchmarks
  - UCX machine layer is better than MPI on lower level benchmarks
    - Up to **127%** when comparing Roundtrip_time_for_1D_Arrays_(zero_copy_message_send_api)
  - UCX Machine layer is better than OFI/Verbs on lower level benchmarks
    - Up to **1707%** when comparing Roundtrip_time_for_Groups_(zero_copy_message_send_api)
- Application Benchmarks
  - NAMD (STMV) using UCX machine layer showed **~5%** improvements over Intel MPI machine layer
- UCX machine layer can be beneficial in the following cases:
  - Small/medium messages take noticeable part of application execution time
  - Application uses Charm++ DIRECT_ONESIDED API, which implemented more efficiently in UCX than in MPI machine layer

Next steps

- Need to finish the merge into Charm++ master branch – work in progress
- Add documentation for how to use it within Charm++ documentation
- Continue investigate and support UCX Machine layer for Charm++
- Test other applications using UCX machine layer



Thank You