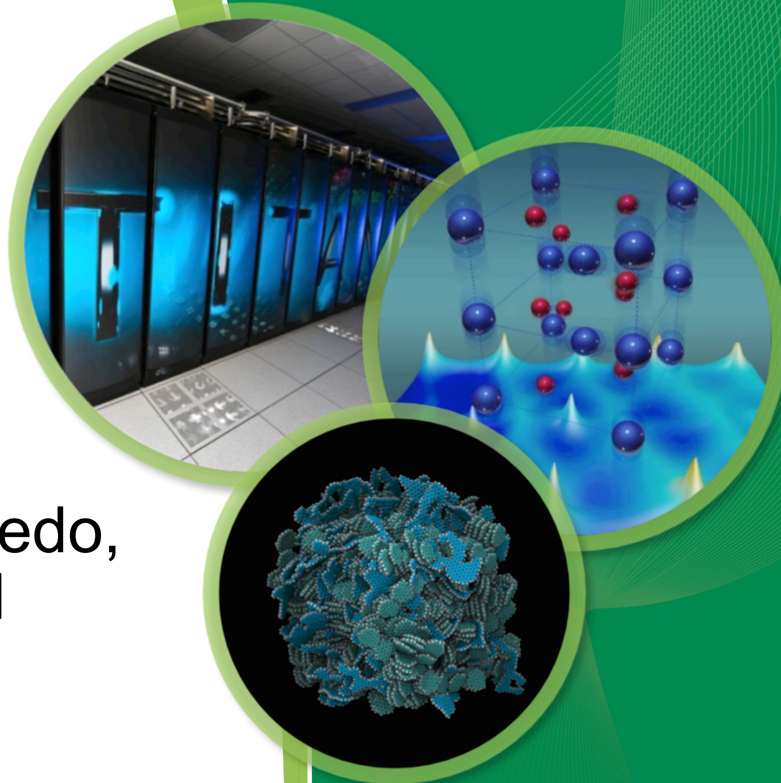
Investigating Data Motion Power Trends to Enable Power-Efficient OpenSHMEM Implementations



Tiffany M. Mintz, Eduardo D'Azevedo, Manjunath Gorentla Venkata, and Chung-Hsing Hsu

2016 OpenSHMEM Workshop

August 3, 2016

Motivation

- Becoming increasingly necessary to be mindful and more in control of power consumed by extreme-scale systems, specifically for data movement
- Lack of software implementations that support or enable power efficient data movement
- Potential to increase power efficiency for memory accesses through power-aware development of OpenSHMEM implementations

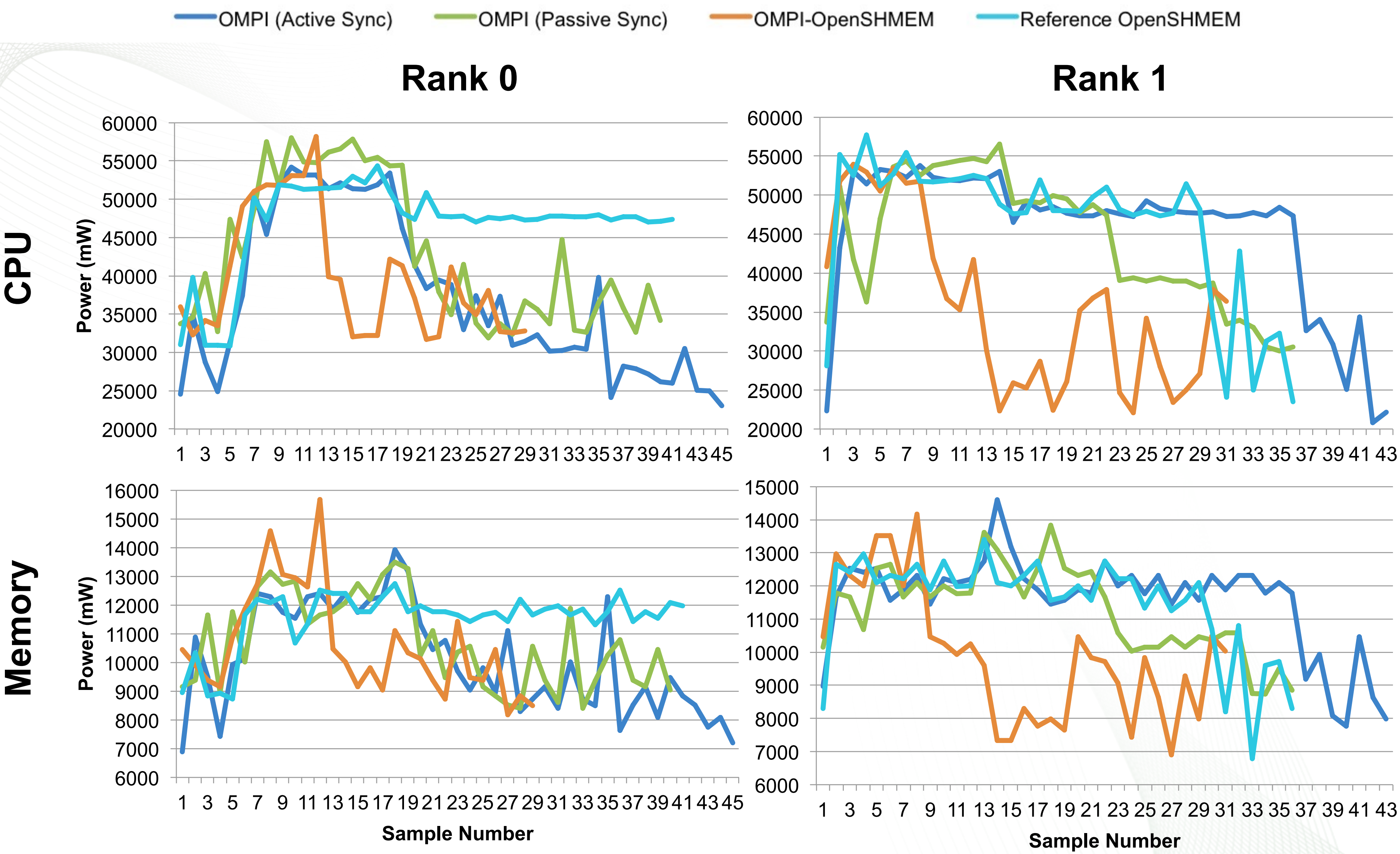
Research Approach

- Current: study power consumption of one-sided RMA operations
  - Profile power consumption for put and get operations for OpenMPI and OpenSHMEM implementations
  - Analyze profiles for significant deviations in power consumption
  - Generate targeted hypothesis for reducing power consumption
- Next: Isolate algorithms within one-sided message passing implementations that could be optimized for power

Power Profiling

- Profiles generated using PowerInsight instrumented cluster
  - Dual Intel Xeon E5-2650v2 i7, 8 cores, 16 threads, a base frequency of 2.6 GHz, and 64GB DDR3-1600 SDRAM
- Benchmarks
  - Ohio State University Micro-Benchmark Suite
    - OpenSHMEM and one-sided MPI put and get latency benchmarks
  - OpenSHMEM implementation of High Performance Conjugate Gradient (HPCG) Benchmark

OSU Micro-Benchmark: Put Operations



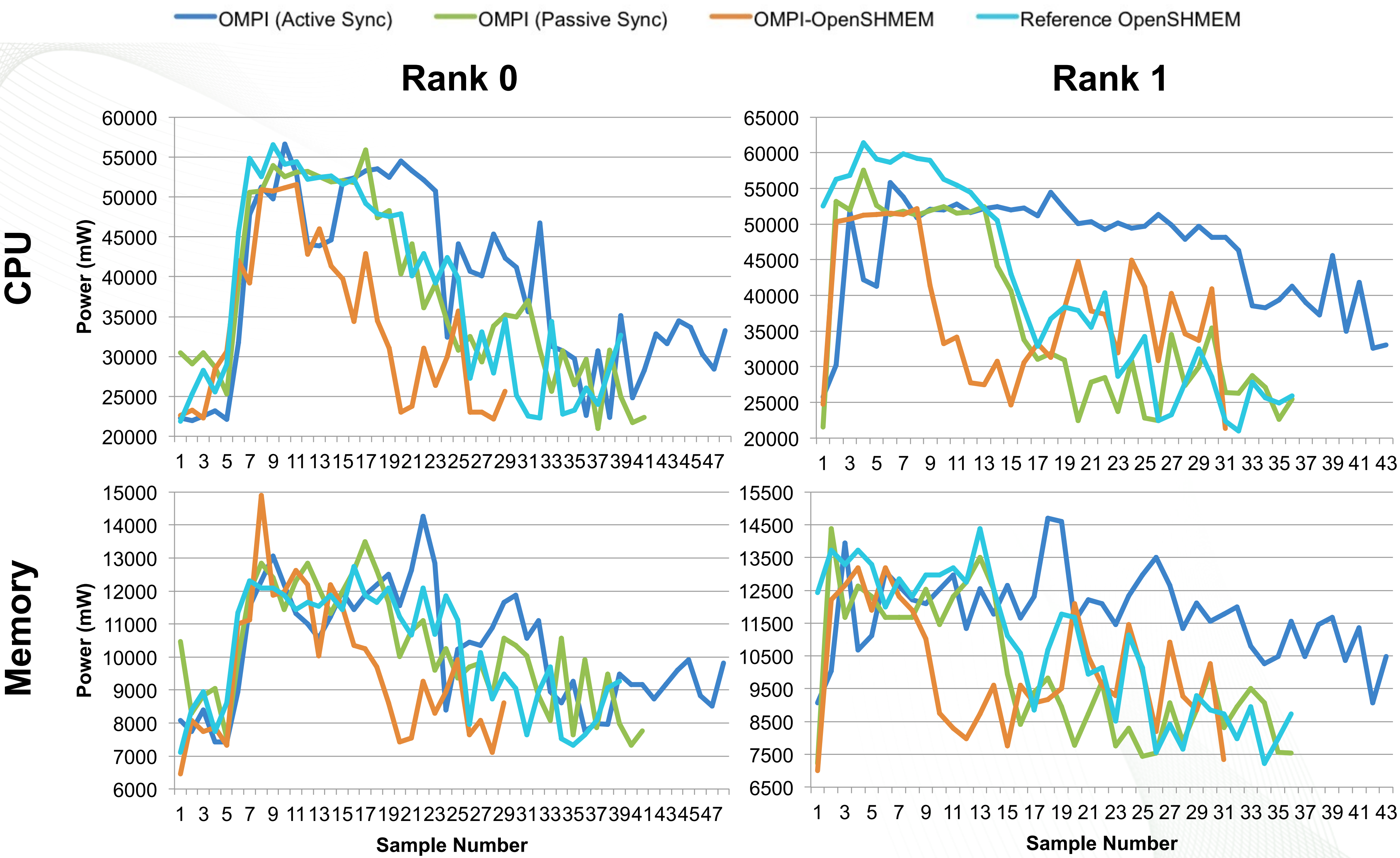
 Computational Research and Development Programs

Oak Ridge National Laboratory

Put Benchmark Observations

- OpenSHMEM Reference implementation has a consistently higher power profile on active process (rank 0) than all other one-sided implementations
  - CPU: On average consumes ~16W – 11W more power, and ~ 63J – 33J more energy
  - Memory: On average consumes ~ 2W more power, and ~ 14J – 3J more energy
- On passive process (rank 1)
  - OpenSHMEM Reference implementation on average consumes more power & energy than the OpenMPI-OpenSHMEM implementation (~12W & 22J cpu, ~2W & 3J memory)
  - OpenSHMEM Reference has comparable power consumption to OpenMPI with active synchronization (< 1W) but consume much less energy (~12J)

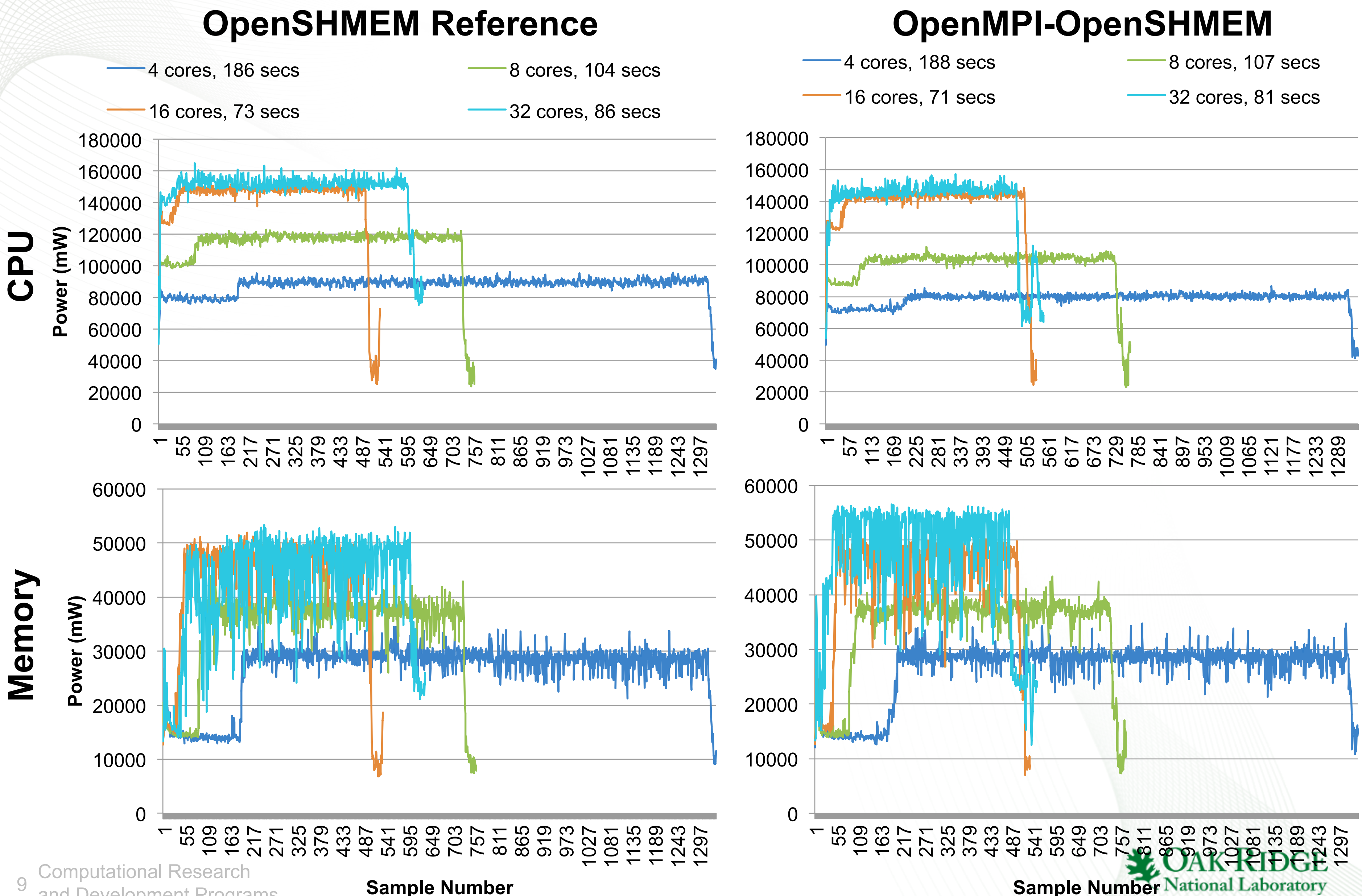
OSU Micro-Benchmark: Get Operations



Get Benchmark Observations

- The OpenMPI-OpenSHMEM implementation on average consume less power and energy on the active process
  - CPU: ~9W – 5W less power, ~125J – 63J less energy
  - Memory: < 2W less power, ~31J – 16J less energy
- On passive process:
  - OpenSHMEM reference implementation consumes less power than OpenMPI-OpenSHMEM (~5W cpu, < 0.5W memory) but consumes more energy (~12J cpu,~ 6J memory)

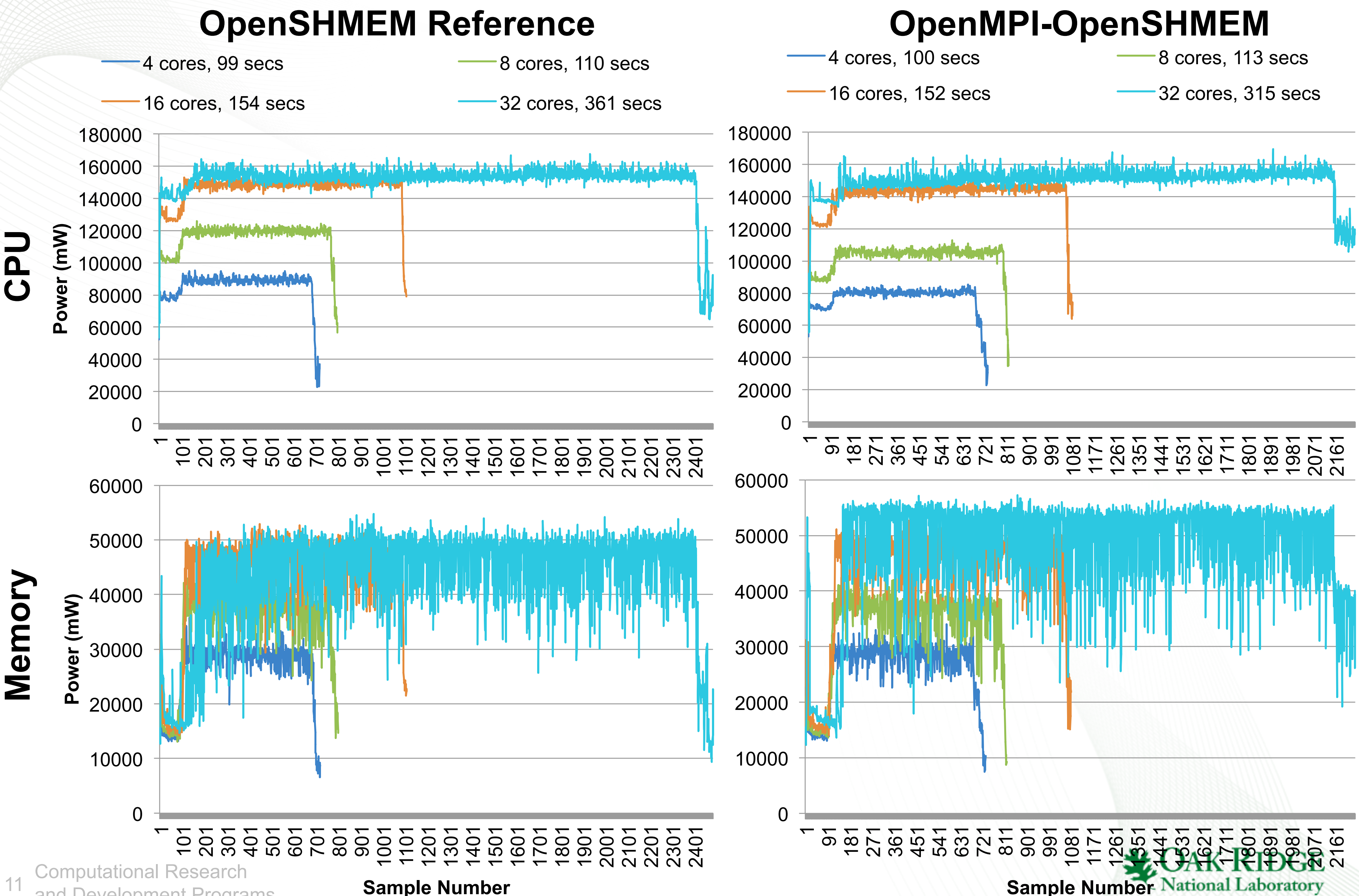
HPCG: Strong Scaling



HPCG Observations

- For both implementations:
  - Non hyper-threaded executions (4-16 cores):
    - delta for cpu power consumption doubles as the number of physical processing cores doubles
    - peak memory power consumption is nearly equivalent across implementations and delta for peak memory power remains constant at ~10W as the number of physical cores doubles
  - For hyper-threaded executions of 32 processes, cpu power consumption nearly equivalent to 16 processes
- Memory power consumption for OpenSHMEM Reference implementation for 32 processes nearly equivalent to 16
  - OpenMPI-OpenSHMEM peak memory power for 32 processes increases by ~ 5W
- OpenSHMEM reference implementation has a peak power profile of about 9W more than the OpenMPI-OpenSHMEM

HPCG: Weak Scaling



Hypothesis from Analysis

- There is not a one-to-one mapping of performance to power consumption in message passing implementations, particularly for memory accesses
- There is a threshold for performance optimizations directly correlating with power optimizations (especially when considering hyper-threaded executions)
- A less power efficient implementation may be optimized for power without degrading performance

Next Steps

- Add power profiles for OpenSHMEM over UCX
- Isolate put and get implementations and determine algorithmic differences in implementations that contribute to disparity in power consumption
- Determine if software re-engineering of put and get operations would result in increase power-efficiency
- Study synchronization models

Acknowledgements

This work was supported by the United States Department of Defense (DoD) and used resources of the Computational Research and Development Programs at Oak Ridge National Laboratory.

Questions?