Job Startup at Exascale:
Challenges and Solutions

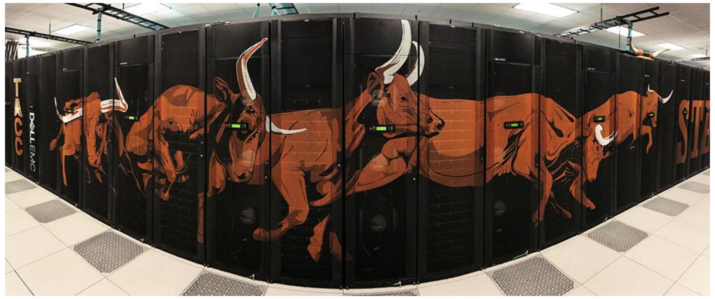
Hari Subramoni

Network Based Computing Laboratory

The Ohio State University

Current Trends in HPC

- Supercomputing systems scaling rapidly
    - Multi-/Many-core architectures
    - High-performance interconnects
- Core density (per node) is increasing
    - Improvements in manufacturing tech
    - More performance per watt
- Hybrid programming models are popular for developing applications
    - Message Passing Interface (MPI)
    - Partitioned Global Address Space (PGAS)



Stampede2 @ TACC

Sunway TaihuLight

Fast and scalable job-startup is essential!

Why is Job Startup Important?

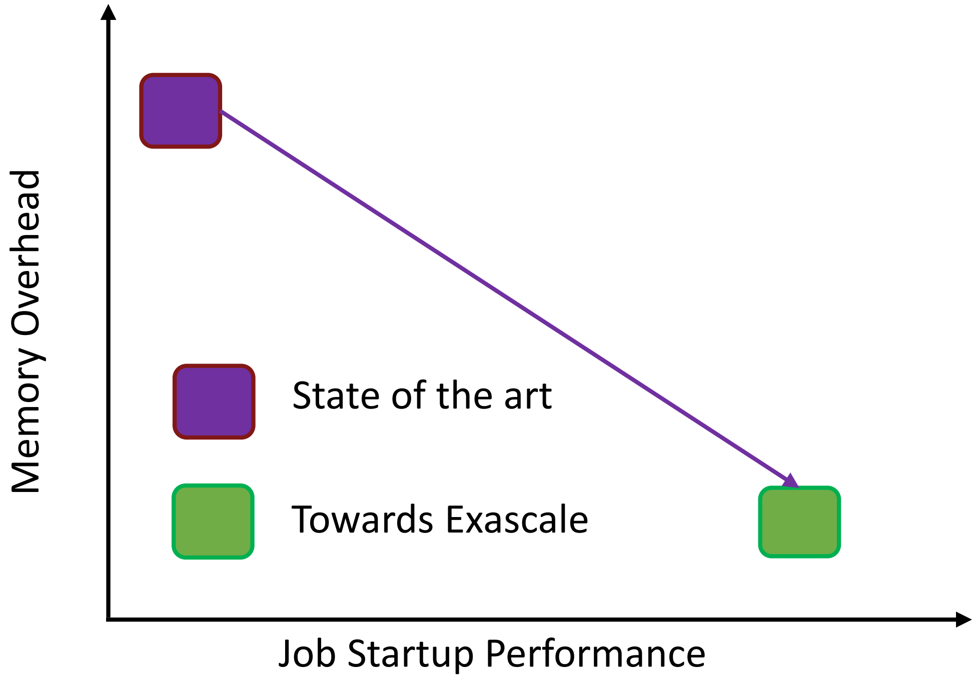
Development and debugging

Regression / Acceptance testing

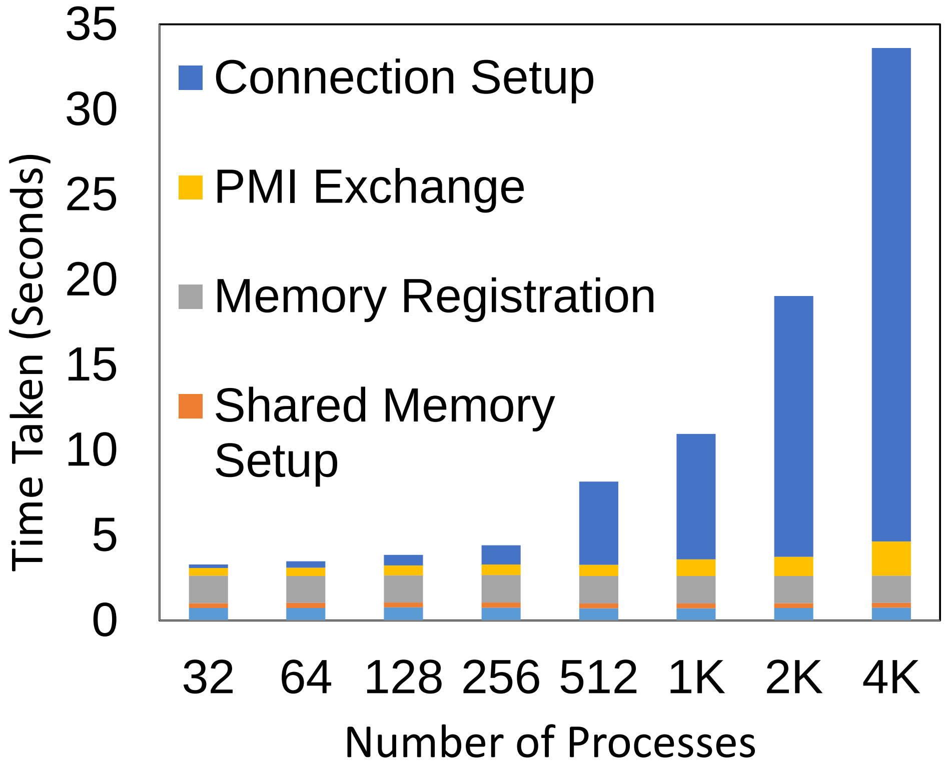
Checkpoint - Restart

Towards Exascale: Challenges to Address

- Dynamic allocation of resources
- Leveraging high-performance interconnects
- Exploiting opportunities for overlap
- Minimizing memory usage



Challenge: Avoid All-to-all Connectivity

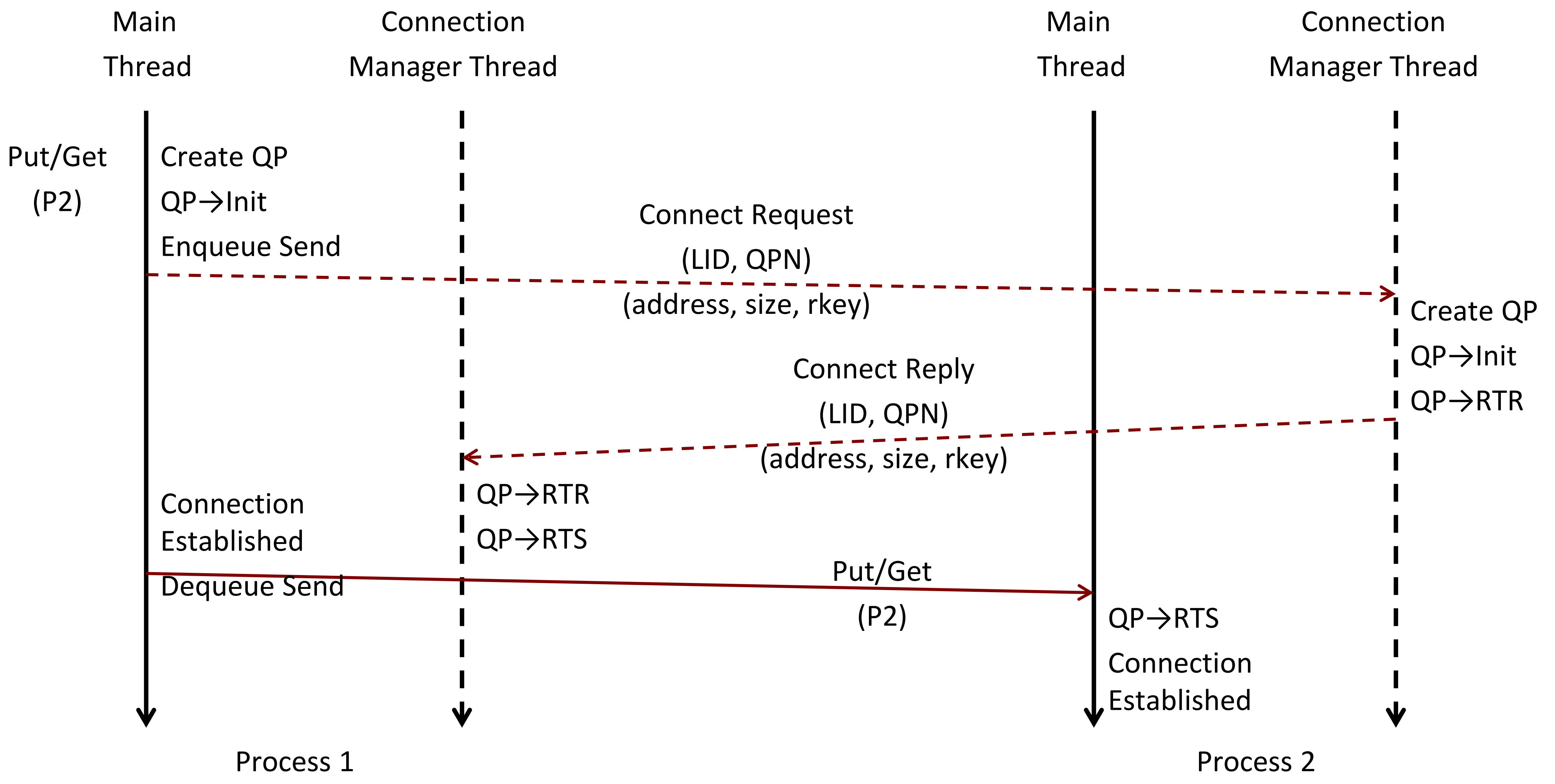


Application	Processes	Average Peers
BT	64	8.7
	1024	10.6
EP	64	3.0
	1024	5.0
MG	64	9.5
	1024	11.9
SP	64	8.8
	1024	10.7
2D Heat	64	5.3
	1024	5.4

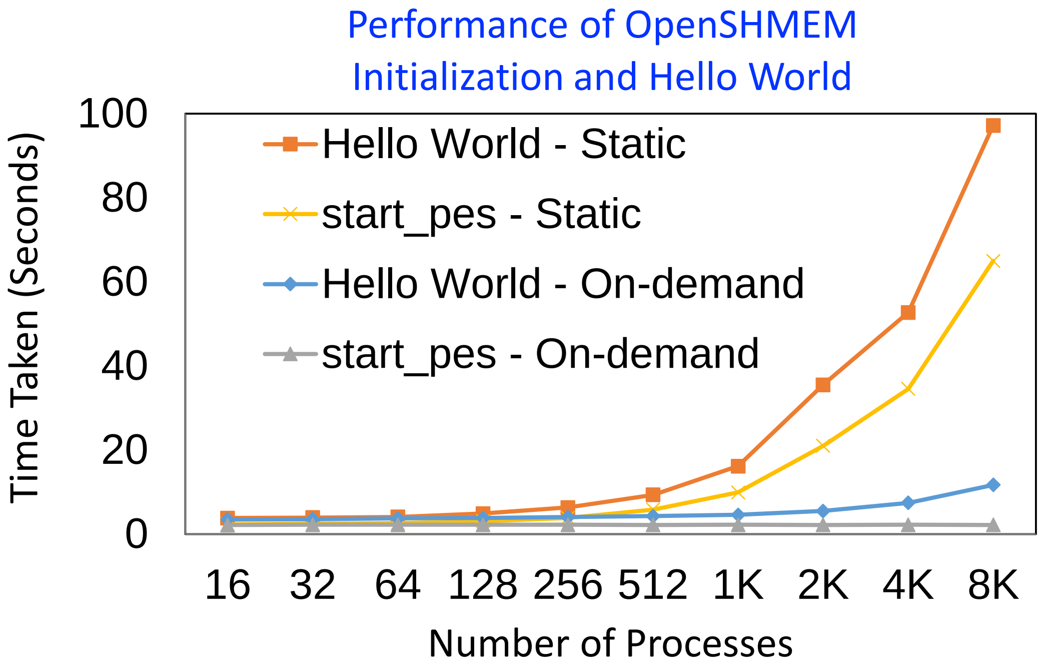
Connection setup phase takes 85% of initialization time with 4K processes

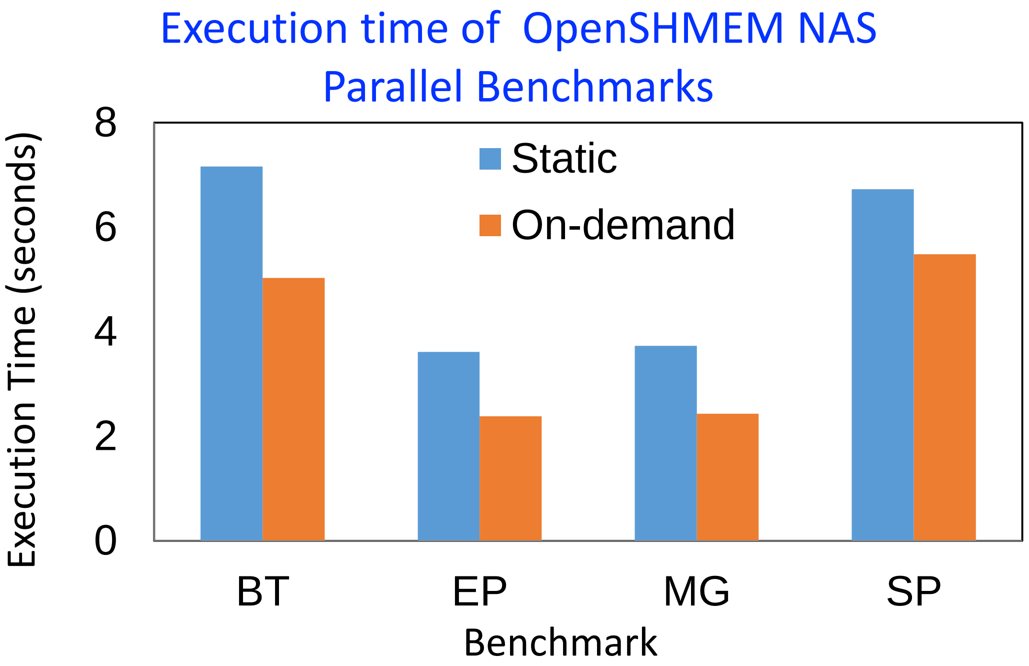
Applications rarely require full all-to-all connectivity

On-demand Connection Management



Results - On-demand Connections



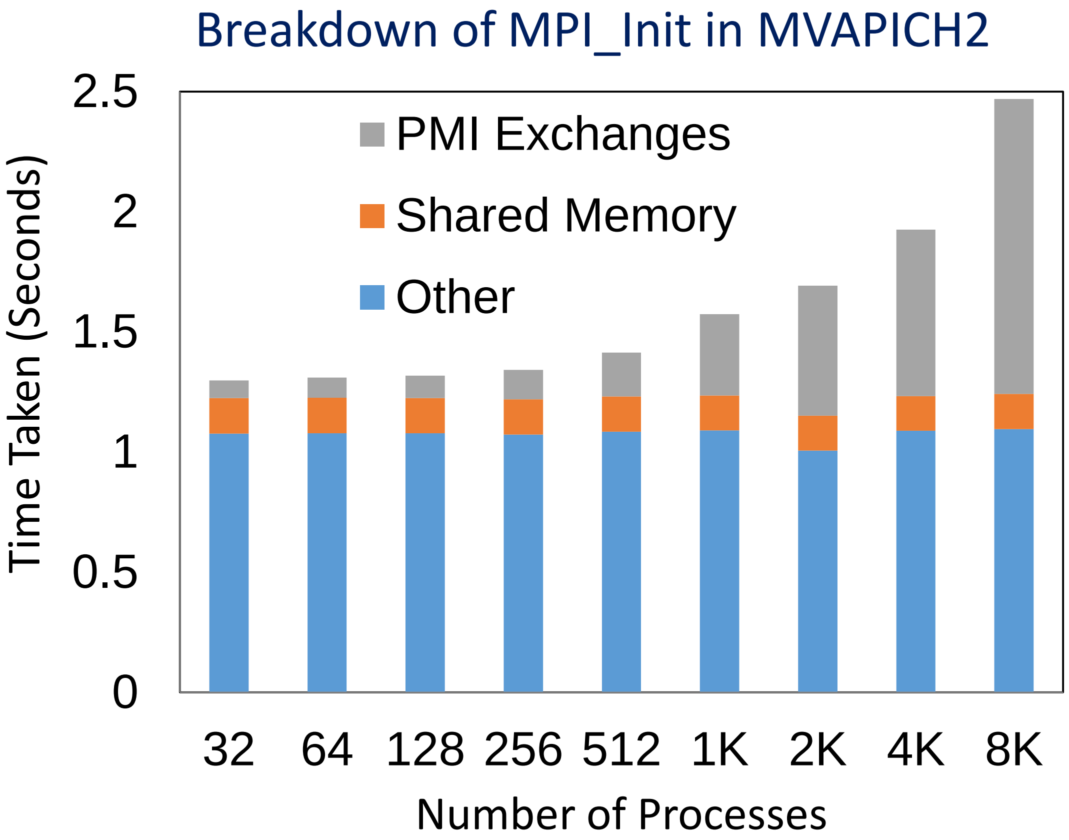


Initialization – 29.6 times faster

Total execution time – 35% better

On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI. S. Chakraborty, H. Subramoni, J. Perkins, A. A. Awan, and D K Panda, 20th International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS ’15)

Challenge: Exploit High-performance Interconnects in PMI



- Used for network address exchange, heterogeneity detection, etc.
  - Used by major parallel programming frameworks

- Uses TCP/IP for transport
  - Not efficient for moving large amount of data
  - Required to bootstrap high-performance interconnects

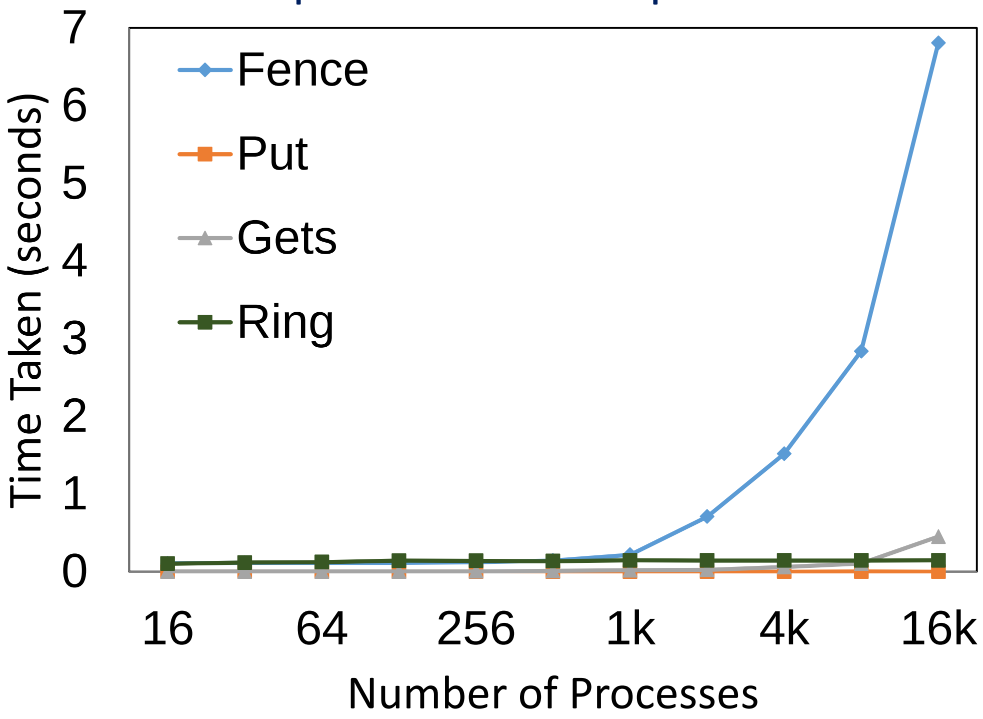
PMI = Process Management Interface

PMIX_Ring: A Scalable Alternative

- Exchange data with only the left and right neighbors over TCP
- Exchange bulk of the data over High-speed interconnect (e.g. InfiniBand, OmniPath)

```
int PMIX_Ring(
    char value[],
    char left[],
    char right[],
…)
```

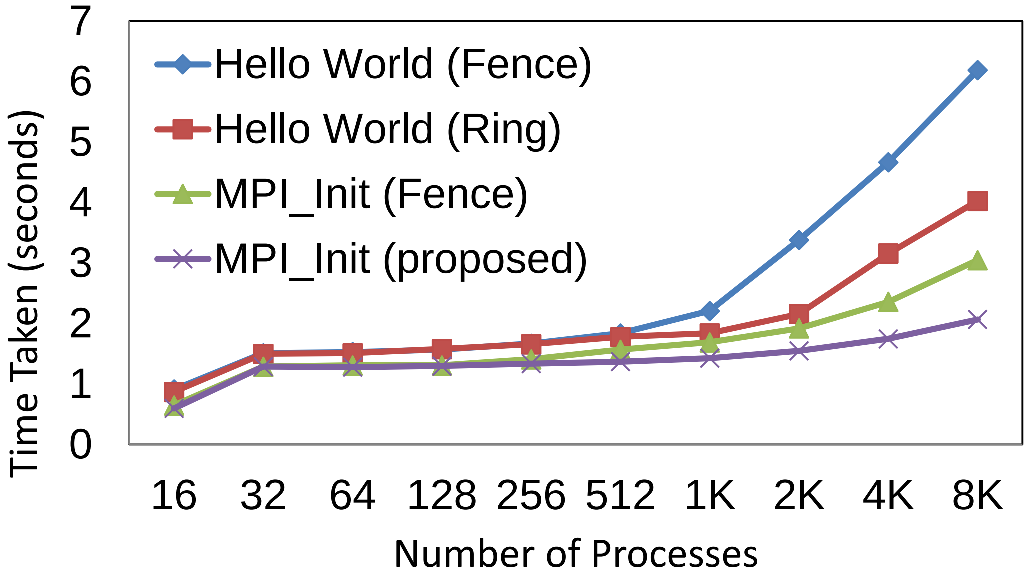
Comparison of PMI operations



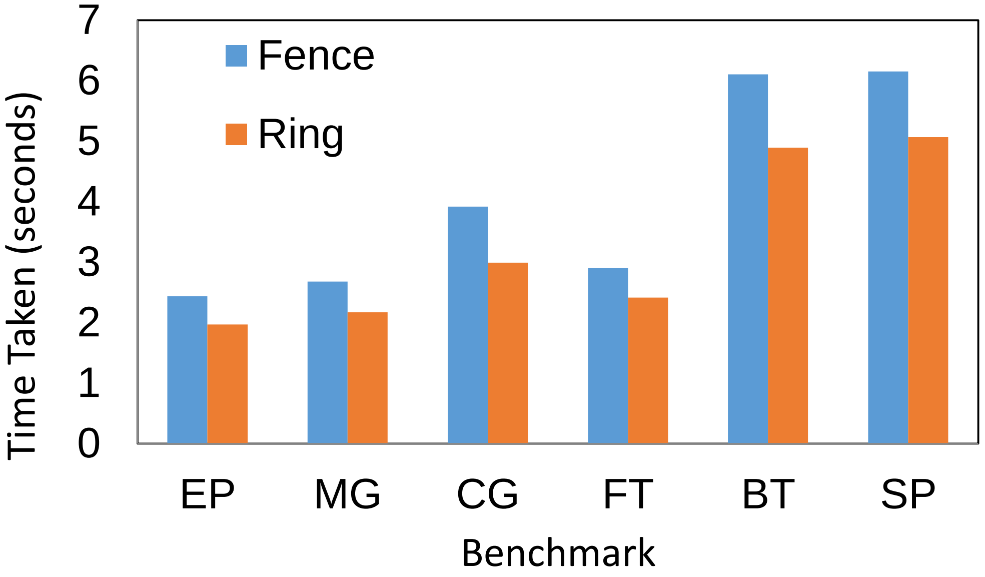
PMIX_Ring is more scalable

Results - PMIX_Ring

Performance of MPI_Init and Hello World with PMIX_Ring



NAS Benchmarks with 1K Processes, Class B Data



33% improvement in MPI_Init

Total execution time – 20% better

PMI Extensions for Scalable MPI Startup. S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia '14)

Challenge: Exploit Overlap in Application Initialization

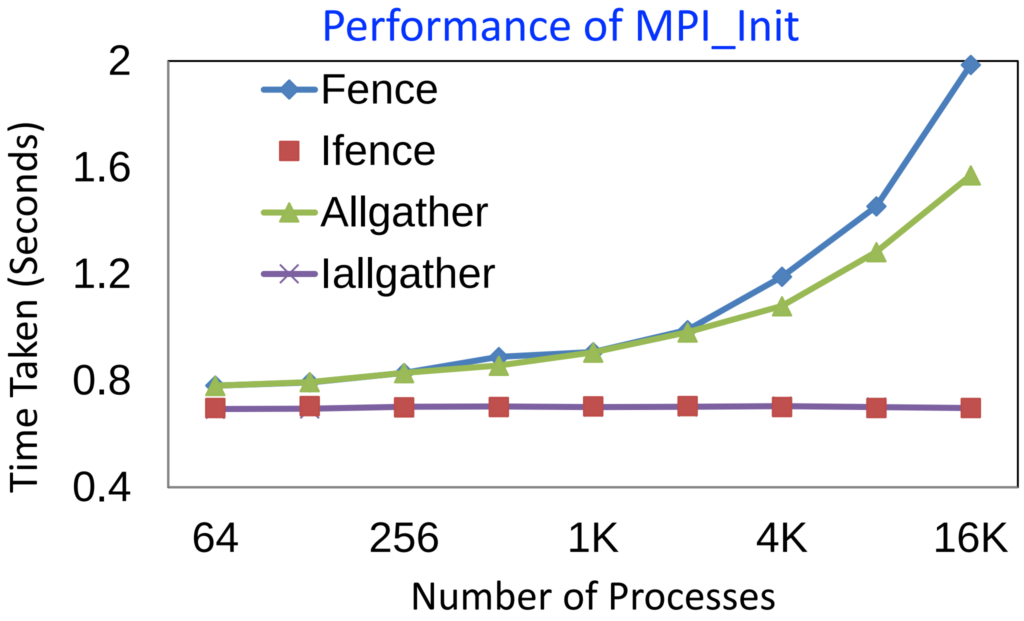
- PMI operations are progressed by the process manager
- MPI/PGAS library is not involved
- Can be overlapped with other initialization tasks / application computation
- Put+Fence+Get combined into a single function - Allgather

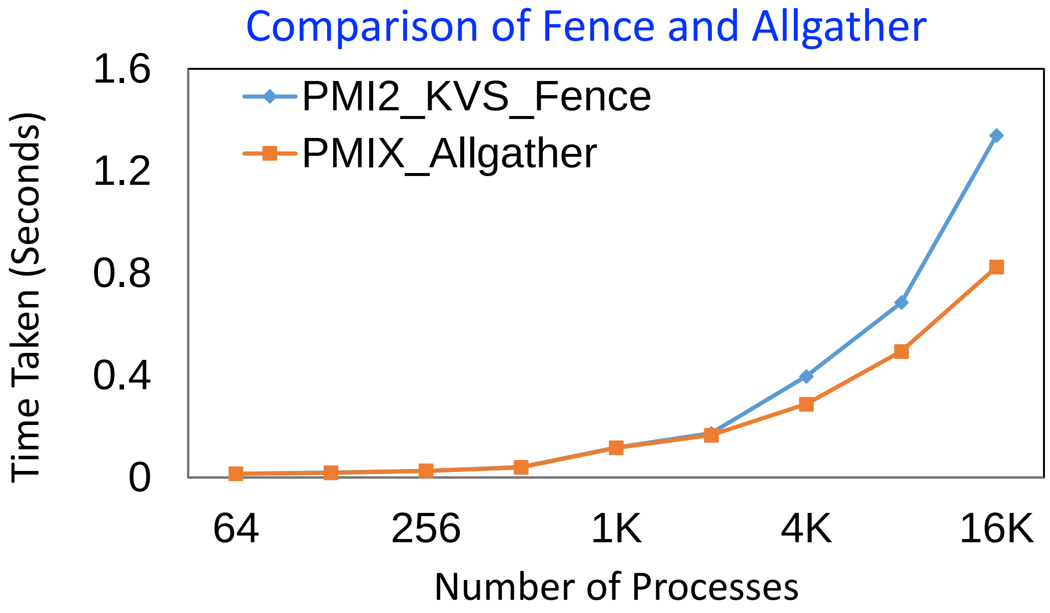
```
int PMIX_KVS_Ifence(
    PMIX_Request *request)

int PMIX_Iallgather(
    const char value[],
    char buffer[],
    PMIX_Request *request)

int PMIX_Wait(
    PMIX_Request request)
```

Results - Non-blocking PMI Collectives





Near-constant MPI_Init at any scale

Allgather is 38% faster than Fence

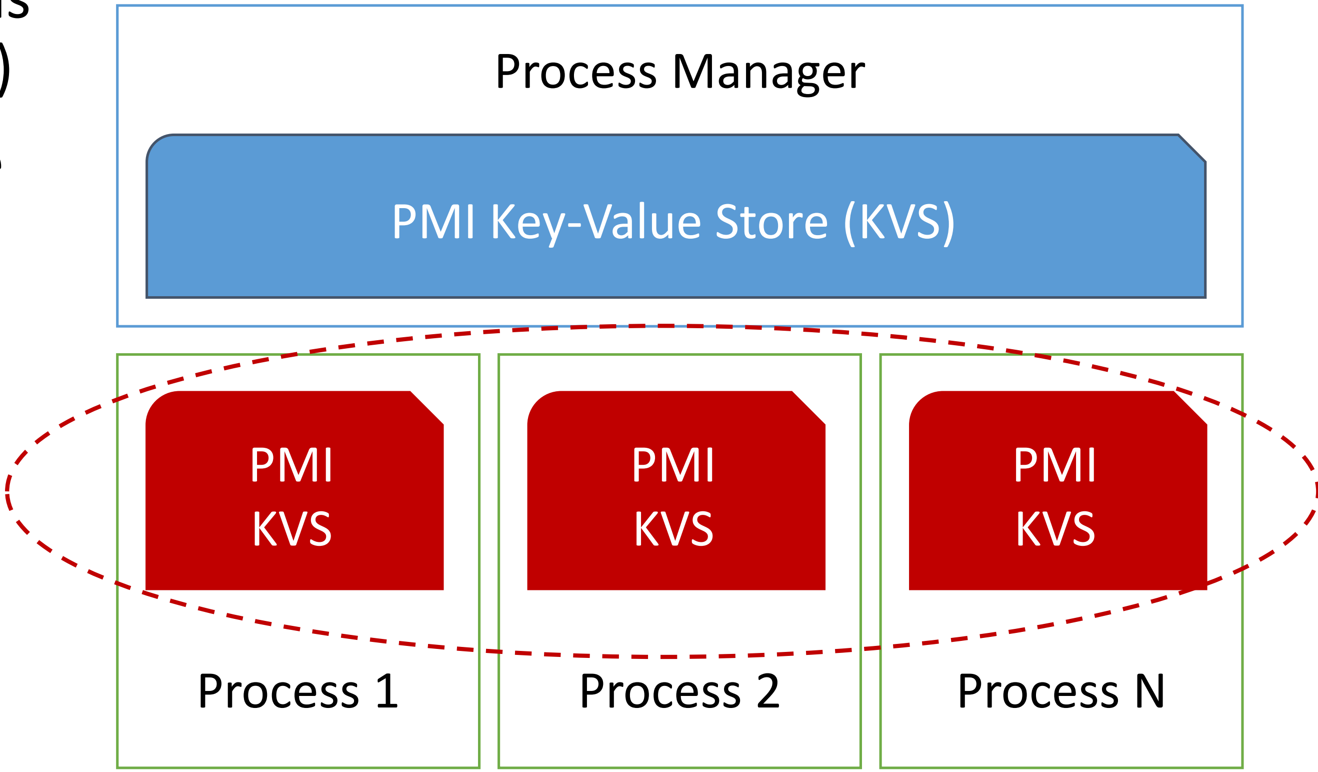
Non-blocking PMI Extensions for Fast MPI Startup. S. Chakraborty, H. Subramoni, A. Moody, A. Venkatesh, J. Perkins, and D K Panda, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid ’15)

Challenge: Minimize Memory Footprint

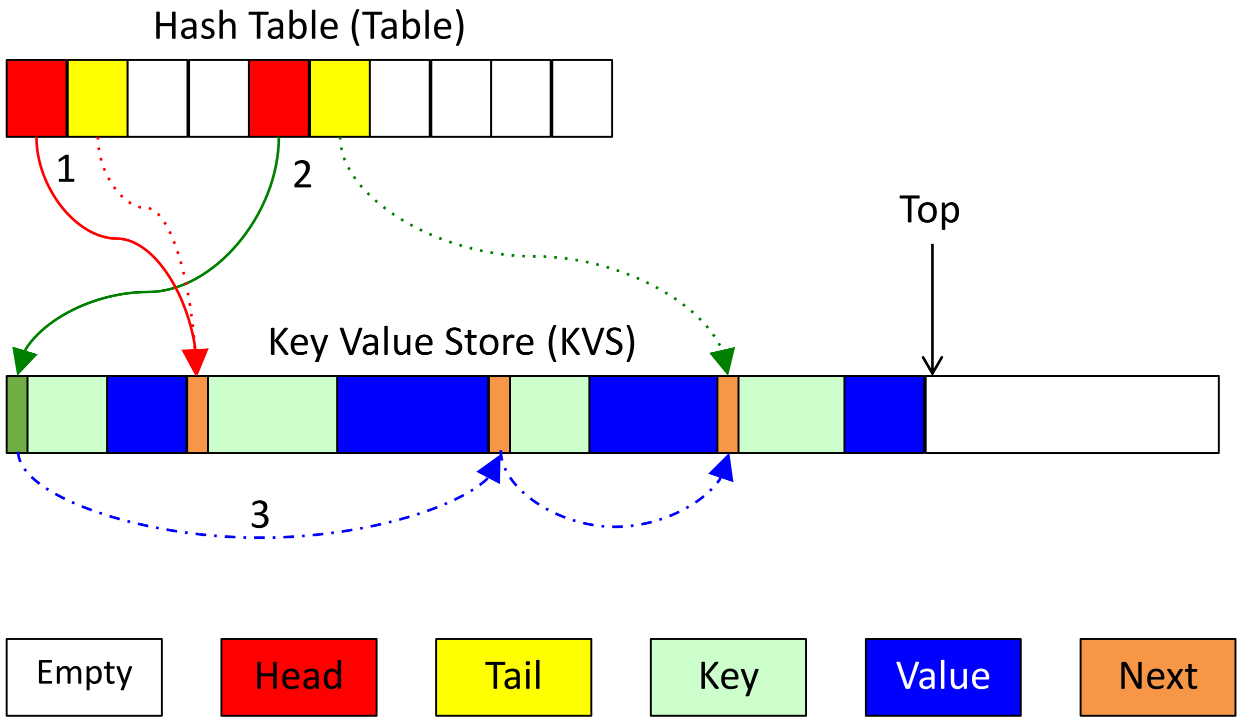
- Address table and similar information is stored in the PMI Key-value store (KVS)
- All processes in the node duplicate the KVS

- PPN redundant copies per node

PPN = Number of Processes per Node

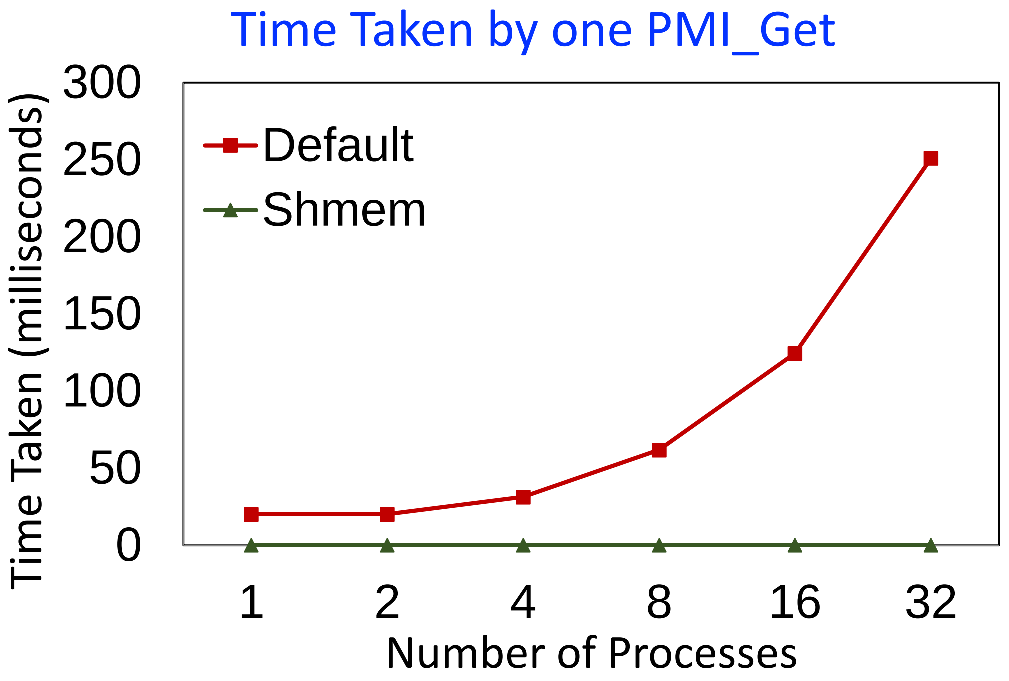


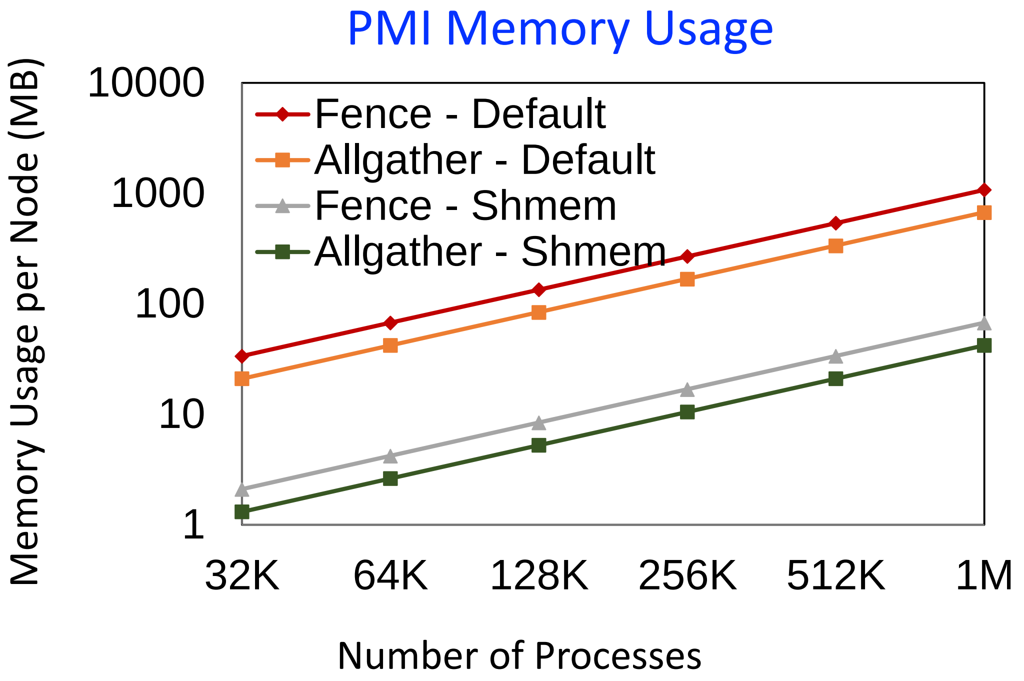
Shared Memory based PMI



- Process manager creates and populates shared memory region
- MPI processes directly read from shared memory
- Dual shared memory region based hash-table design for performance and memory efficiency

Shared Memory based PMI



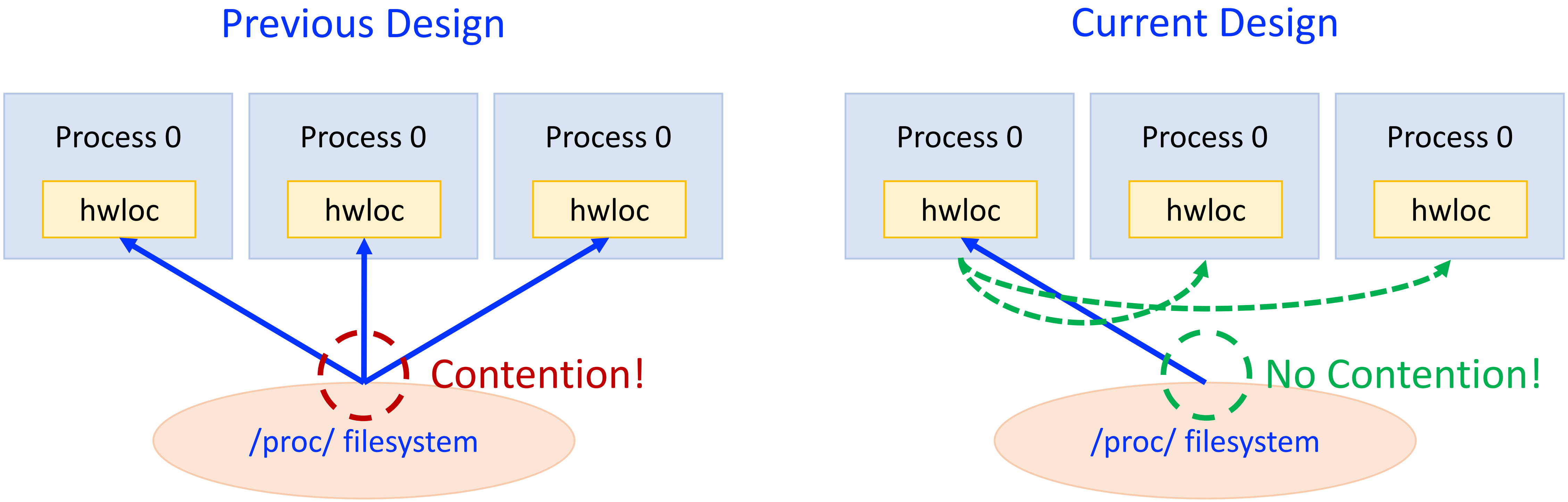


PMI Gets are 1000x faster

Memory footprint reduced by O(PPN)

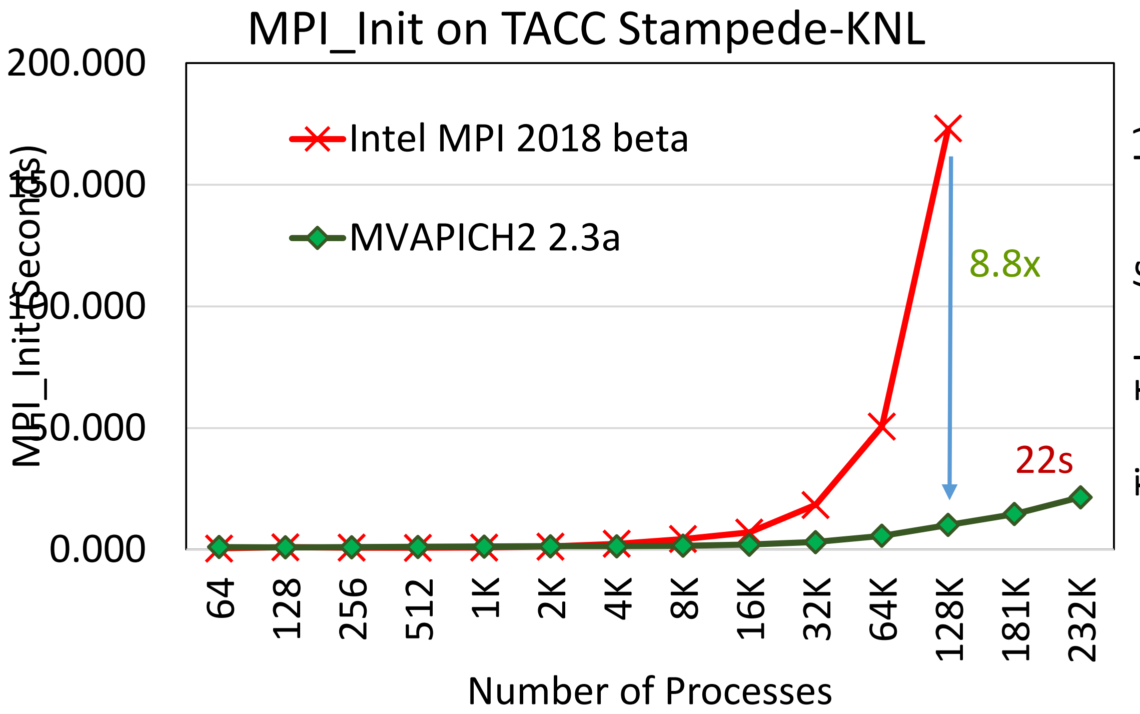
SHMEMPMI – Shared Memory based PMI for Improved Performance and Scalability. S. Chakraborty, H. Subramoni, J. Perkins, and D K Panda, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '16)

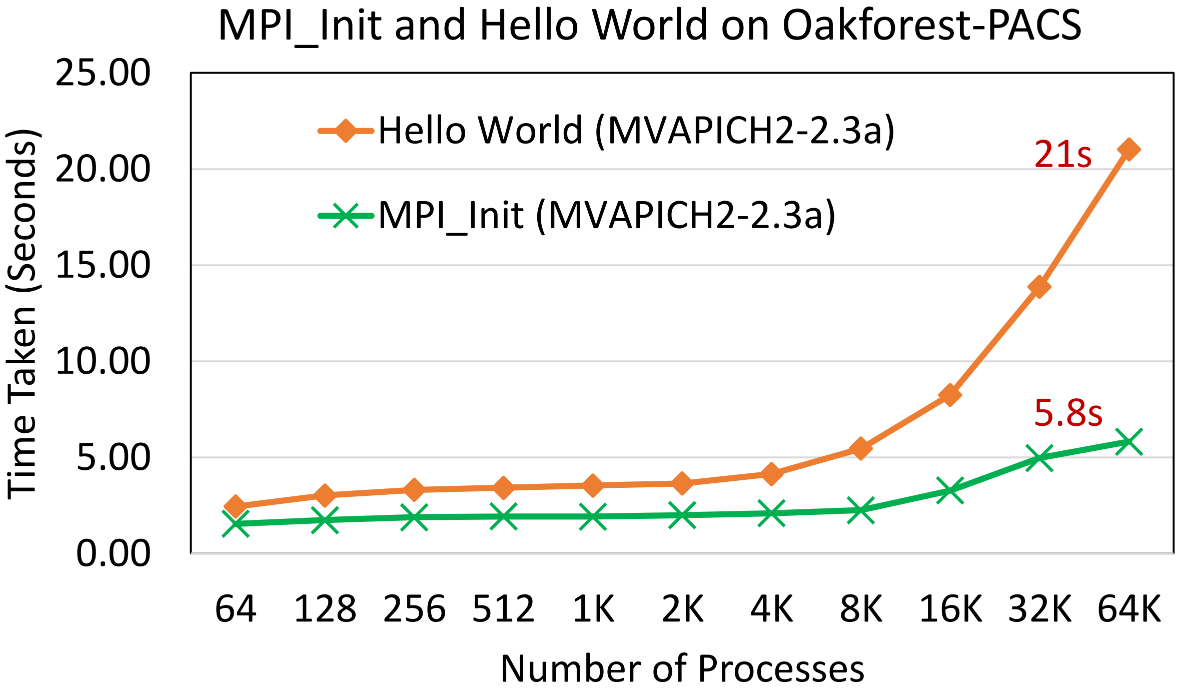
Efficient Intra-node Topology Discovery



Significant improvement on Many-core systems

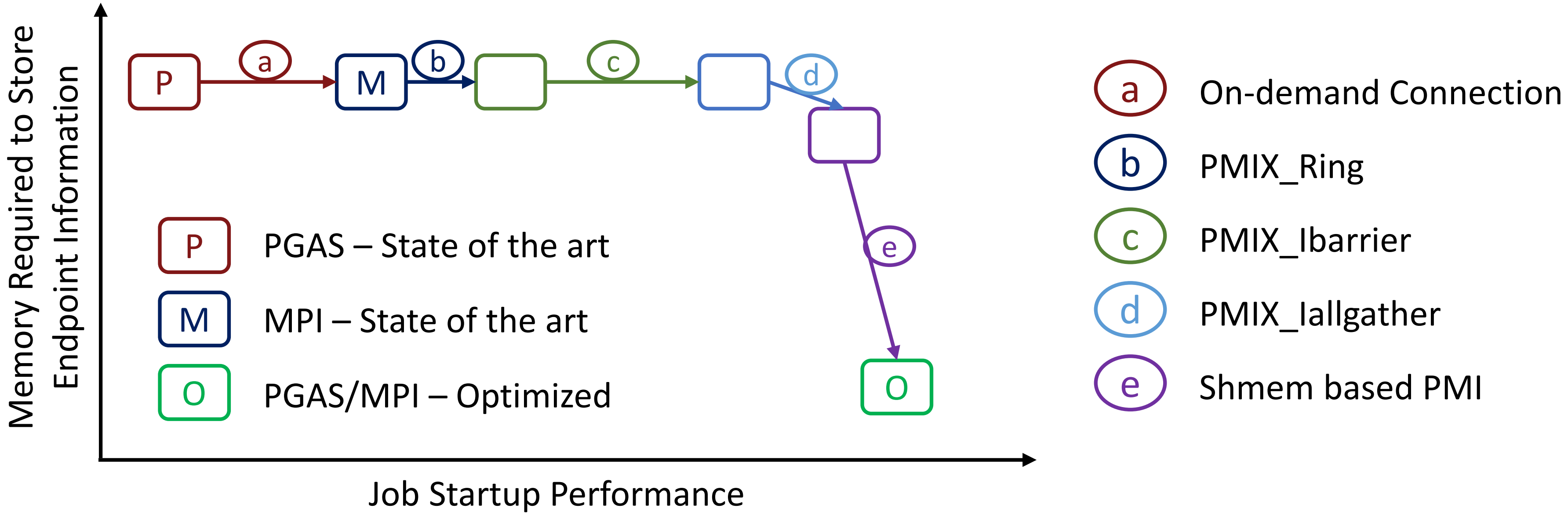
Startup Performance on KNL + Omni-Path





- MPI_Init takes 22 seconds on 231,956 processes on 3,624 KNL nodes (Stampede – Full scale)
- 8.8 times faster than Intel MPI at 128K processes (Courtesy: TACC)
- At 64K processes, MPI_Init and Hello World takes 5.8s and 21s respectively (Oakforest-PACS)
- All numbers reported with 64 processes per node

Summary



- Near constant MPI/OpenSHMEM initialization at any process count
- 10x and 30x improvement in startup time of MPI and OpenSHMEM with 16,384 processes (1,024 nodes)
- Full scale startup on Stampede2 under 22 seconds with 232K processes
- O(PPN) reduction in PMI memory footprint

Optimized designs available in MVAPICH2 and MVAPICH2X-2.3b

Thank You!

http://go.osu.edu/mvapich-startup

http://mvapich.cse.ohio-state.edu/

subramon@cse.ohio-state.edu

panda@cse.ohio-state.edu