OpenSHMEM Workshop 2015

Check-pointing approach for fault tolerance in OpenSHMEM

Pengfei Hao
Pavel Shamis,
Swaroop Pophale,
Tony Curtis,
Barbara Chapman

Fault Tolerance

Necessary:

- HPC system consist more components than before.
- Mean Time To Failure(MTTF) of individual components failed to increase.

Previous Related Work

N.Ali, S.Krishnamoorthy, N.Govind, and B.Palmer. **A redundant communication approach to scalable fault tolerance in PGAS programming models.** In *Parallel Distributed and Network-Based Processing(PDP) , 2011 19th Euromicro International Conference*.

Limitation:

- Algorithm specific solution. “Maintaining and continuously updating shadow data structures.”
- Manually effort to manage backup and checkpoint restore

Previous Related Work

Hao, Pengfei, et al. **Fault Tolerance for OpenSHMEM**. *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models. ACM, 2014*.

Goal:

- Unified Model for fault tolerance
- Unified API for check-pointing, restore.

Design target

Fault mitigation. Not fault detector.
Assuming exist a perfect fault detector.

OpenSHMEM Memory Model

Symmetric Regions:

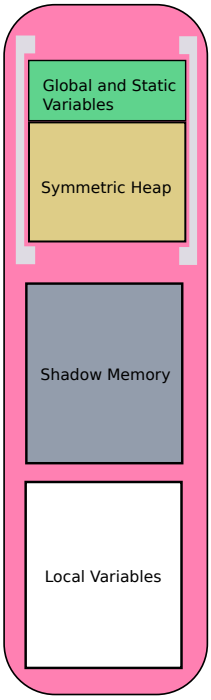
- Fortran data object in common blocks or with the SAVE attribute
- Global and Static C and C++ variables
- Fortran mem allocated by shpalloc()
- C and C++ mem allocated by shmalloc()

User need to store algorithm data in symmetric region.

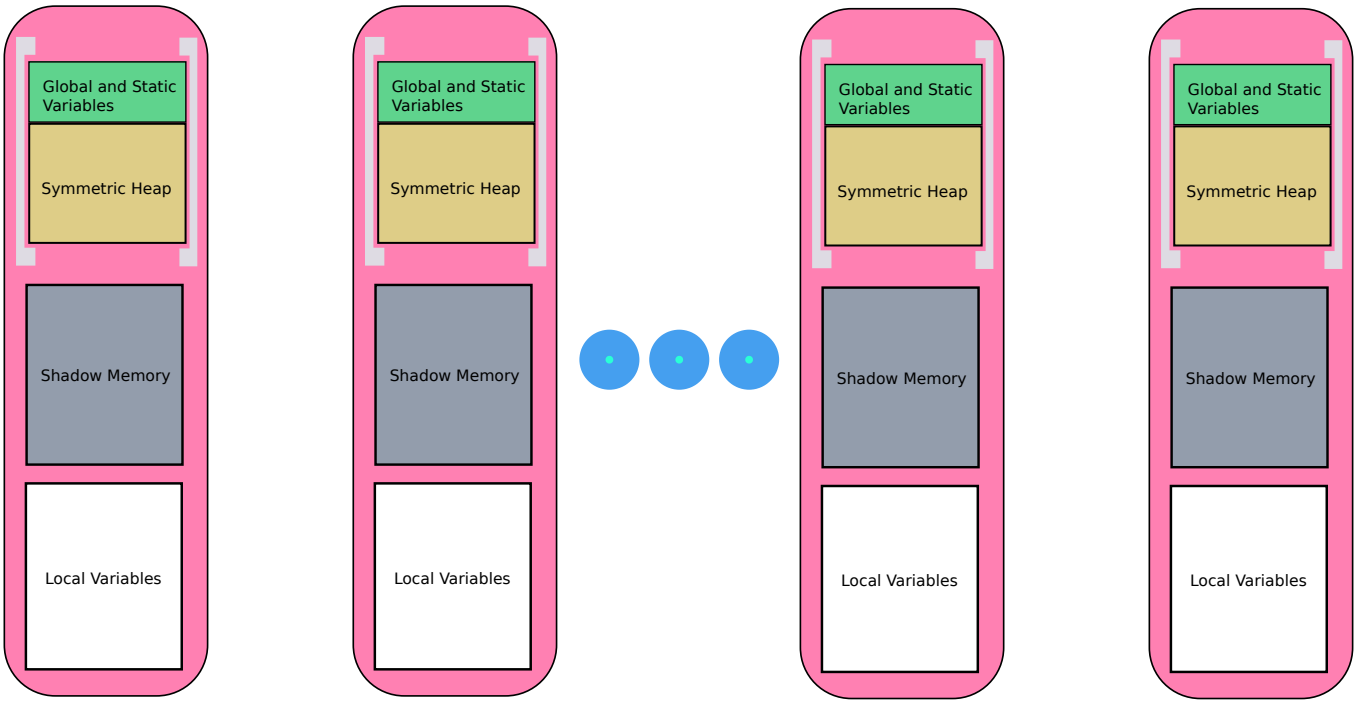
Fault tolerance Design

- User Level Fault Mitigation(ULFM)
  User need to call APIs to support the features.
- Backup all symmetric regions
  For ELF, BSS/DATA/HEAP
- Substitute PE capacity
  Spawn new or sleep as substitute
- Partner PE and shadow memory
  Multiple symmetric regions support

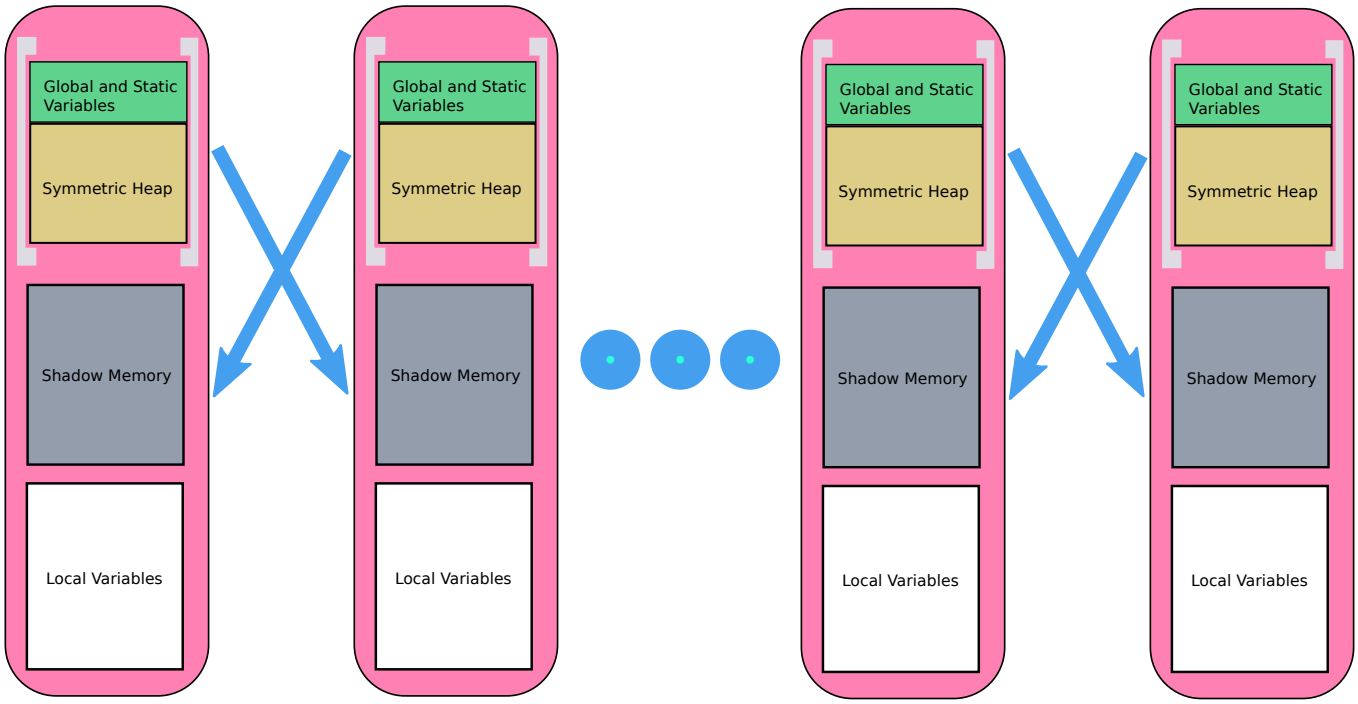
Fault tolerance Design - Memory Layout



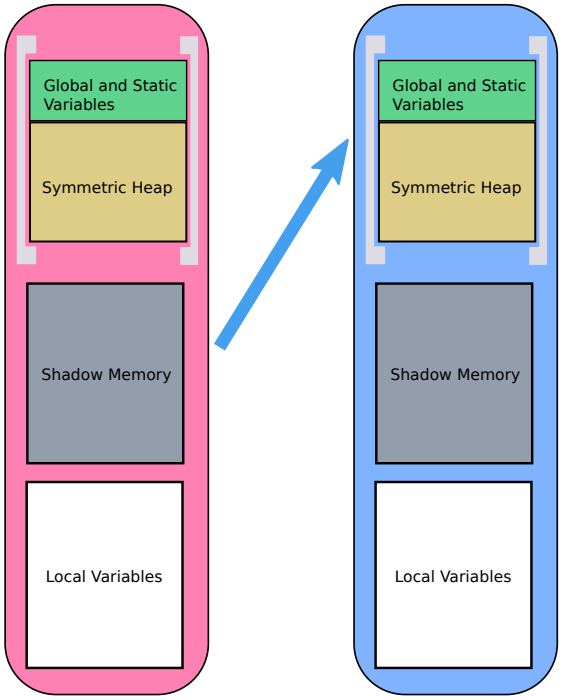
Fault tolerance Design - Backup



Fault tolerance Design - Backup



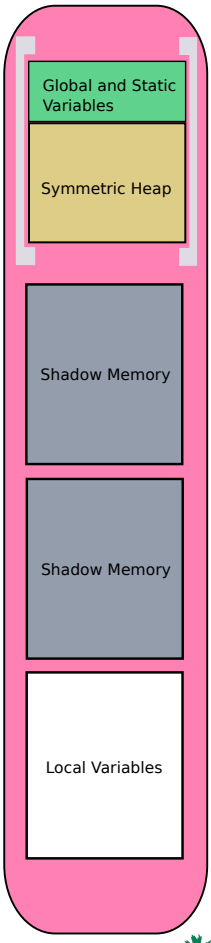
Fault tolerance Design - Restore



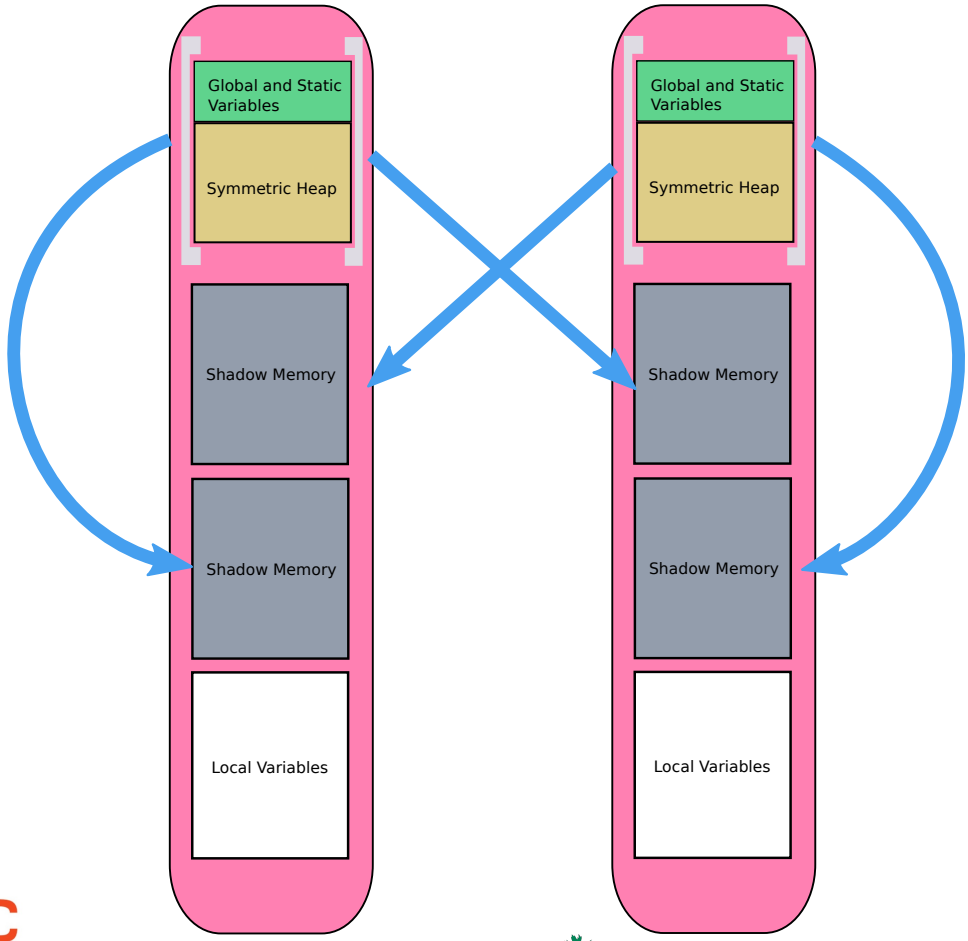
Restore selection

- PE fails – Checkpoint of partner lose
- Single backup VS. Dual backup

Dual backup


Global and Static Variables
Symmetric Heap
Shadow Memory
Shadow Memory
Local Variables

Dual backup


Global and Static Variables
Symmetric Heap
Shadow Memory
Shadow Memory
Local Variables
Global and Static Variables
Symmetric Heap
Shadow Memory
Shadow Memory
Local Variables

Single Backup VS. Dual backup

- Dual backup
  - Local restore benefit
  - Fully consistency
- Single backup
  - Better memory usage
  - Fit for some algorithms

API

- int shmem_checkpoint_all(void);
  Fault detection by return value. Return false indicates faliure happens. Always return false for substitute PE
- int shmem_query_fault(int **pes, int **pes_status, size_t *numpes);
  Check failed PE list.
- int shmem_restart_pes(int *pes, size_t num_pes);
  Collective reconstuction of PE to process mapping.
- int shmem_ft_algo_init(void);
  Guard the initialization code to avoid initialization stages on substitute PEs.

User Level Fault Mitigation

```
#include <shmem.h>
/* User algorithm states as global variables */
int step;
int main(int argc, const char *argv[])
{
    start_pes(0);
    if(shmem_ft_algo_init()){
        /* User algorithm initialization goes here */
    }
    do{
        if(SHMEM_FT_SUCCESS!=shmem_checkpoint_all()){
            int *pes, *pes_status;
            size_t numpes;
            shmem_query_fault(&pes,&pes_status,&numpes);
            shmem_restart_pes(pes,&numpes);
            shmem_barrier_all();
            free(pes);
            free(pes_status);
        }
        else{
            /* User algorithm computing loop goes here*/
            step++;
        }
    }while(step<MAX_STEP);
    /*User algorithm result report goes here*/
    return 0;
}
```

Implementation

- Based on UCCS
- Currently targetting spec 1.0
- Role changing
  Substitute PE change role from sub to work. Within shmem_restart_pes
- Process to PE remapping
  Substitute PE will take failed PE's PE number, no need to care about PE number change.
- Modules' reset, like shmem_barrier_all() Library/Algorithm initialization.

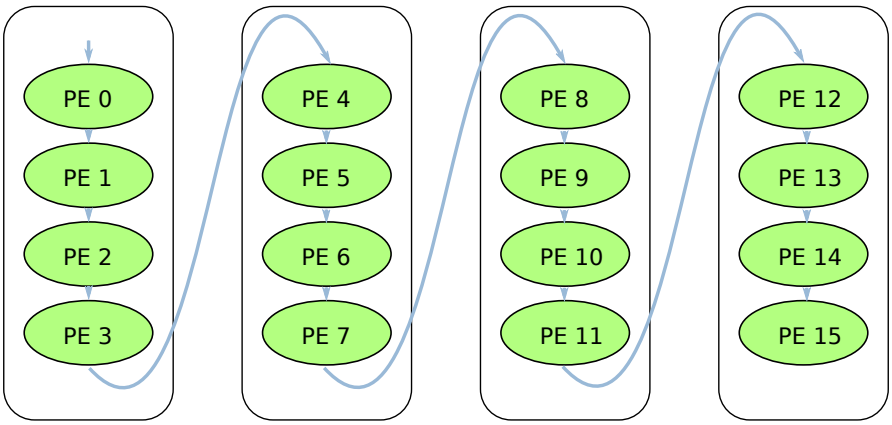
Implementation

Limitations:

- Currently using simple topology
  Current Ring. API do not limit the topology.
  More future work on topology adaption.
- Cannot shmalloc() within computing steps
  Metadata transferable allocator
- Only support dynamic linkage now
  In static, all symbols mingled. Need compiler to solve this.

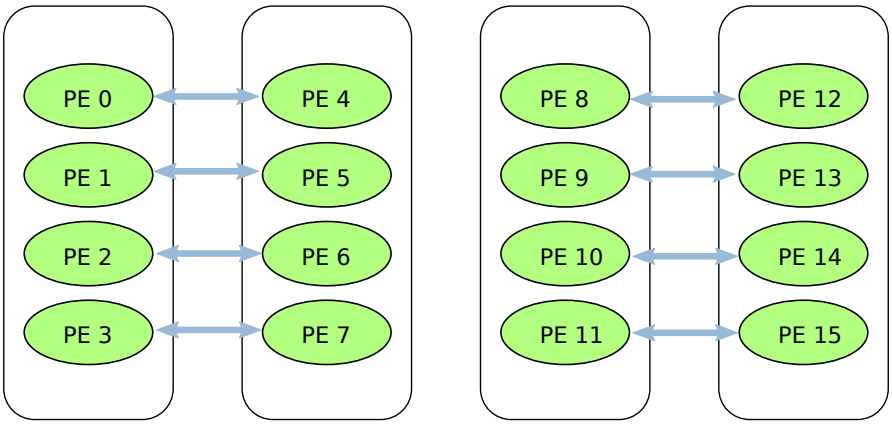
Some more talk about topology

Current implementation, for simplicity



Some more talk about topology

A better topology



Some more talk about topology

What is a good topology for backing up?

- Bandwidth usage
- Fault recovery ability
- Awareness
- Adaption

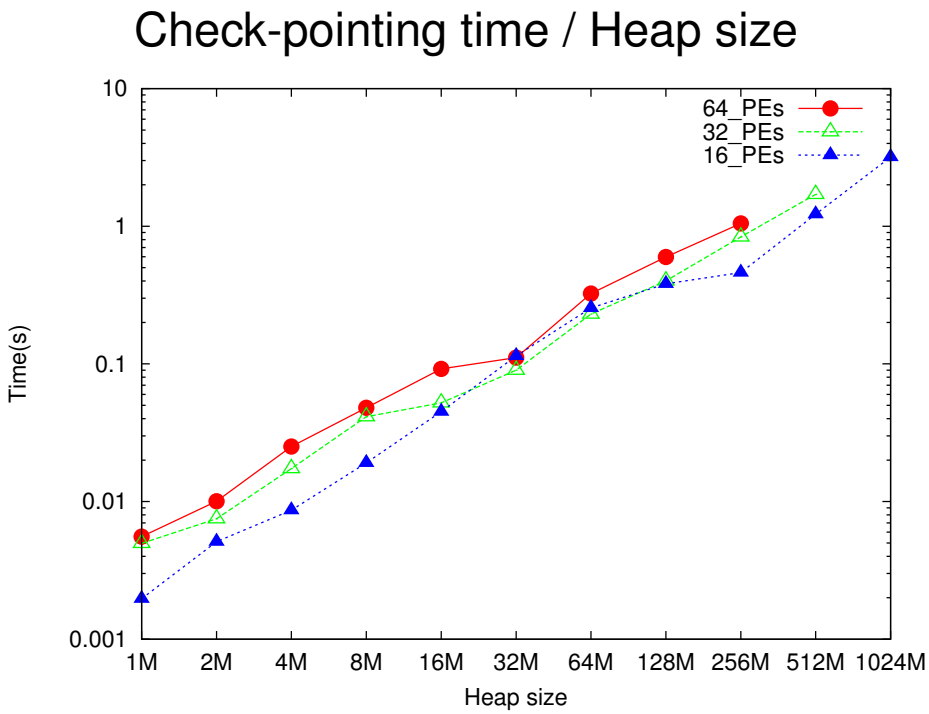
Evaluation

Crill cluster:

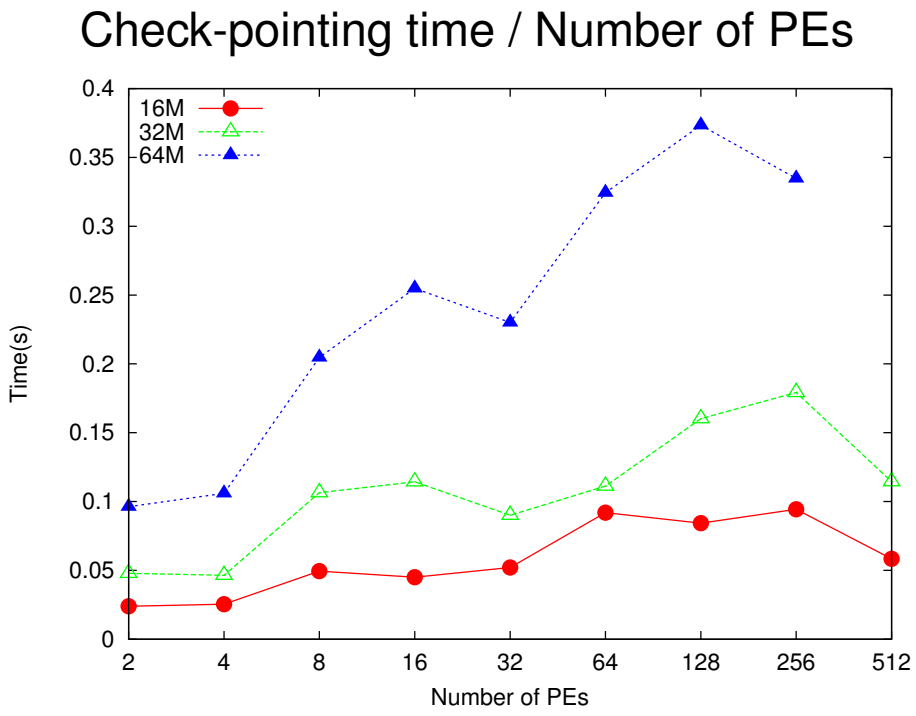
- 16 Nodes
- 2.2GHz AMD Opteron 6174 processor
- 64GB RAM on each node
- dual port Infiniband ConnectX HCA (20GB per second)

Process mapping: -bycore

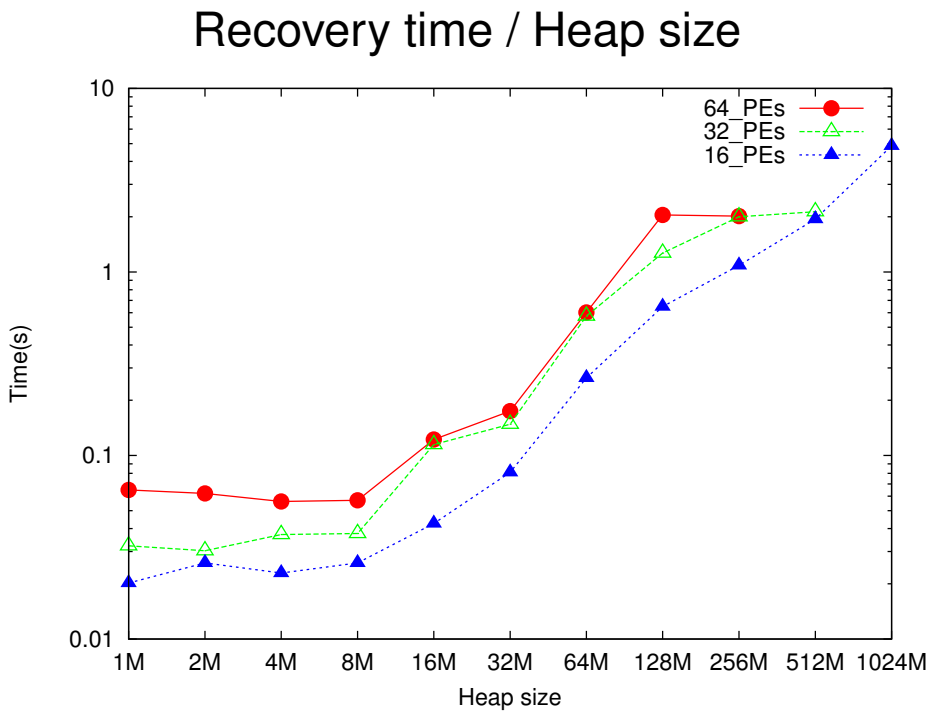
Evaluation


Check-pointing time / Heap size
64_PEs
32_PEs
16_PEs
Time(s)
10
1
0.1
0.01
0.001
1M
2M
4M
8M
16M
32M
64M
128M
256M
512M
1024M
Heap size

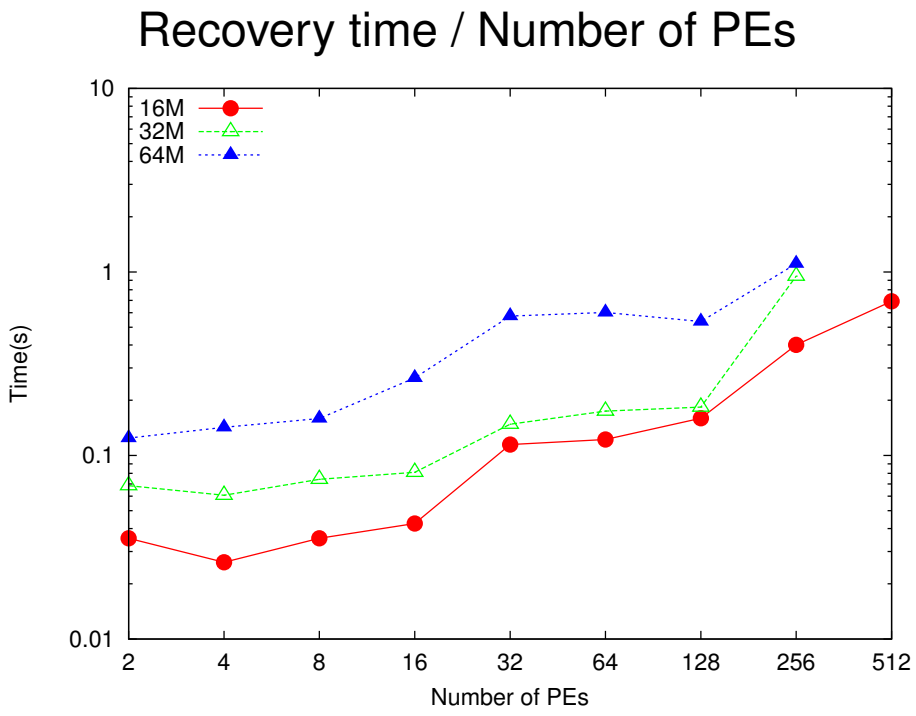
Evaluation


Check-pointing time / Number of PEs
16M
32M
64M
Time(s)
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
2
4
8
16
32
64
128
256
512
Number of PEs

Evaluation

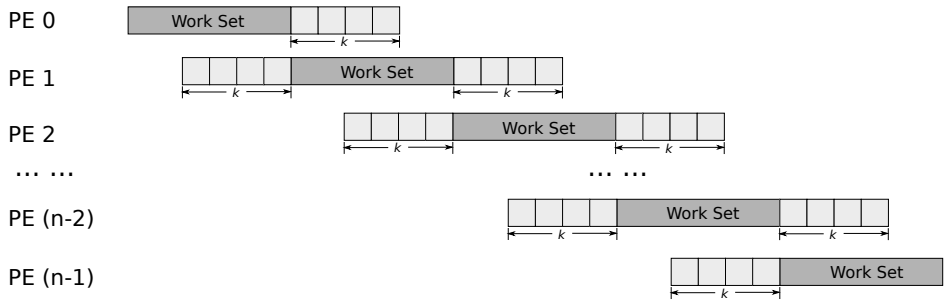

Recovery time / Heap size
Time(s)
10
1
0.1
0.01
1M
2M
4M
8M
16M
32M
64M
128M
256M
512M
1024M
Heap size
64_PEs
32_PEs
16_PEs

Evaluation


Recovery time / Number of PEs
16M
32M
64M
Time(s)
10
1
0.1
0.01
2
4
8
16
32
64
128
256
512
Number of PEs

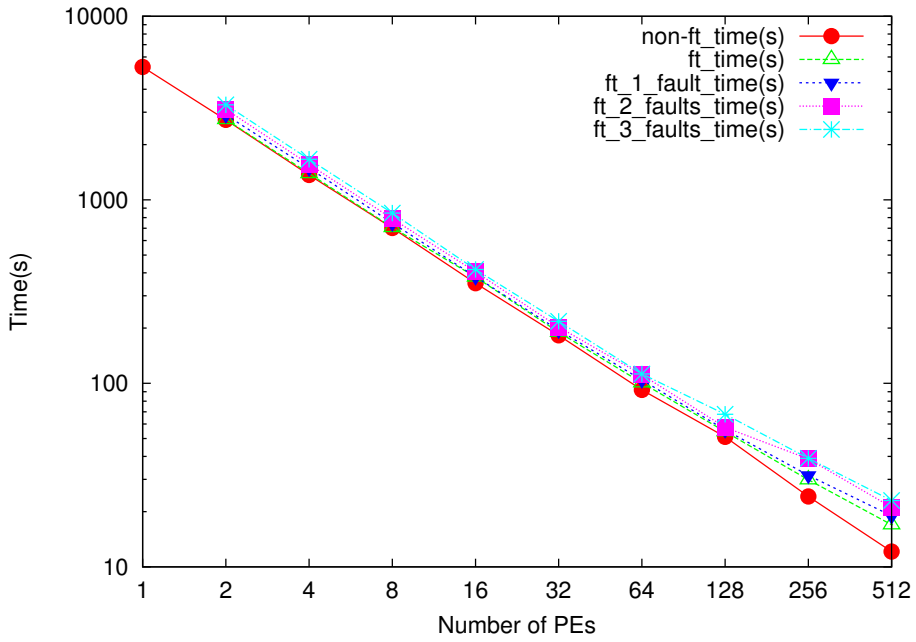
Evaluation

Jacobi 1D Stencil



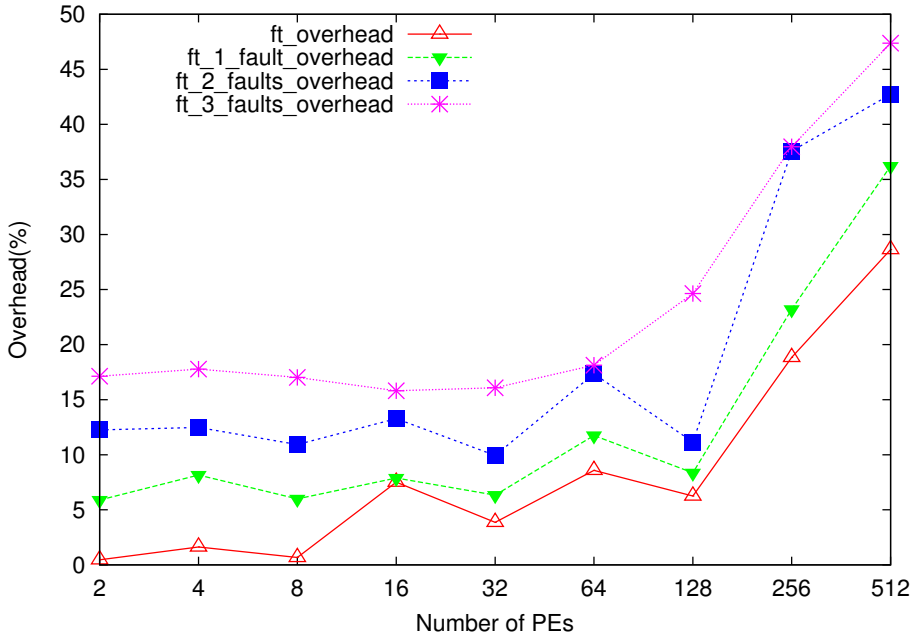
Evaluation

Jacobi 1D, 64M size, 4096 iterations, 16 check-points



Evaluation

Jacobi 1D, 64M size, 4096 iterations, 16 check-points



Summary

- Fault Tolerance ULFM for OpenSHMEM
- APIs
- Backup/Restore Strategy
- Evaluation

Future Work

- New comms layer, UCX
- New spec, catch up 1.2
- Topology Awareness and Adaption
- Combine with resonable fault detector

Acknowledgements

This work was supported by the United States Department of Defense & used resources of the Extreme Scale Systems Center at Oak Ridge National Laboratory.

Thank you!

Questions?

HPC TOOLS

OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY