Integrating Fault Tolerance into the Monte Carlo Application Toolkit

Rob T. Aulwes
Los Alamos National Laboratory
CCS-7
Motivation

- Going to exascale means more hardware failures
- Mean Time To Interrupt (MTTI) goes down and checkpoint time goes up
  - Result: more time creating dump files than doing actual work
  - Up to 45 min to create VPIC restart file
  - Larger dumps also mean longer restarts
  - Why abort a 10,000+ processor job just because 1 process failed?
Goals of Project

- Raise awareness at LANL for need to address failures
  - A large (6000+ PE) cosmology run was attempted, but had difficulties making progress due to multiple failures
  - Fault tolerance is now part of discussions about how to prepare for exascale
  - Part of Level 2 Milestone
  - Press for need of LANL to contribute to developing fault-tolerant OpenMPI

- Demonstrate ability to make a production code fault-tolerant
  - Presented to Monte Carlo Codes group
The Monte Carlo Application ToolKit (MCATK)

- Two-year-old project to write a parallel Monte Carlo neutron transport code using modern software engineering practices
- Supports domain-replicated, domain-decomposed, hybrid
- Domain-replicated provided easiest model to demonstrate fault tolerance
FT-MPI

- Fault-tolerant MPI from U. of Tennessee
- Extends MPI semantics to include fault tolerance
  - Detect if restarted process
  - Get list of failed ranks
- Provides recovery modes: REBUILD, BLANK, SHRINK
- However, no longer being developed or maintained
- Only made aware of failure through MPI call
- Is not integrated with Totalview
Recovery Modes

- **REBUILD**
- **BLANK**
- **SHRINK**
Boost and MPI

- MCATK uses Boost MPI C++ library
- MPI errors translated to Boost MPI exceptions
- Communicators created dynamically and wrapped with shared_ptr
- Communicators become invalid after a failure
  - Used Observer design pattern to design notification system
  - Listeners responded to failures and recreated communicators
Fault-tolerant scheme (SHRINK mode)

- Group MPI ranks into local checkpoint groups
- Each rank in group sends its particles to every other rank within group
- On failure, lowest-ranked processor in group takes over particles of all failed processes within group
  - Have load imbalance for only one cycle
- State rolled back to start of failed cycle
- If only 1 PE remaining in group, then abort
Local Checkpoint Groups

Group 1

Group 2

Group N/4
Local Checkpointing and Recovery

- Particles stored in neighbor’s memory
  - Obviously, very memory intensive strategy
  - However, design allows for storage to local disk as an option
  - Future architectures include advances in non-volatile local storage

- Implemented notification system to notify interested objects about failures
  - Needed to update any reference to Boost MPI communicators

- Recovery does not complete until no more failures
  - Failures also handled if occur during recovery
Recovery Logic - SHRINK

1. Get Failed List
2. Failures in group?
   - Yes: Am I lowest rank in group?
     - Yes: Take over particles
     - No: No
   - No: Restore local particles
3. Rollback to failed cycle
Testing

- Ran a K-effective calculation on 64 Pes
  - K eigenvalue is a measurement of criticality
  - Test has reproducible result

- Tested multiple types of failures
  - Multiple failures
  - Simultaneous
  - Failures within recovery

- However, did experience hangs with some tests
## Test Results

- **Turing and Yellowrail with 64PEs**

<table>
<thead>
<tr>
<th># Particles</th>
<th>20 * 64</th>
<th>200 * 64</th>
<th>500 * 64</th>
<th>1000 * 64</th>
</tr>
</thead>
<tbody>
<tr>
<td>no failures</td>
<td>0.998762397</td>
<td>1.00046942</td>
<td>1.00120465</td>
<td>0.999840359</td>
</tr>
<tr>
<td>3 failures</td>
<td>0.998762397</td>
<td>1.00046942</td>
<td>1.00120465</td>
<td>app hang</td>
</tr>
<tr>
<td>2 simultaneous</td>
<td>0.998762397</td>
<td>1.00046942</td>
<td>1.00120465</td>
<td>0.999840359</td>
</tr>
</tbody>
</table>
Future Work

- Collaborate with current efforts to incorporate FT-MPI features into Open-MPI

- Extend to domain-decomposed
  - Solving domain-decomposed would provide insight to adding fault tolerance to Eulerian codes

- Expand effort to other apps at LANL
  - Eulerian hydrodynamics
  - Radiation transport
  - VPIC