Debugging/Testing Non-deterministic MPI Applications

ECP 2nd annual meeting
February 9th

Kento Sato
What is non-determinism in MPI applications?

- Message receive orders change across executions
  - Unpredictable system noise (e.g. network, system daemon & OS jitter)
- Floating point arithmetic orders can also change across executions

Execution binary + Input data

Execution A: \((a+b)+c\)

Execution B: \(a+(b+c)\)
Non-determinism increases debugging cost

- Control flows of an application can change across different runs

- Non-deterministic control flow
  - Successful run, seg-fault or hang
- Non-deterministic numerical results
  - Floating-point arithmetic is non-associative

\[(a+b)+c \neq a+(b+c)\]

In non-deterministic applications, it’s hard to reproduce bugs and incorrect results. It costs excessive amounts of time for “reproducing” target behaviors.
Non-deterministic bugs cost substantial amounts of time and efforts in MPI applications.

**Diablo/Hypre 2.10.1**
- The bug manifested in particular clusters
- It hung only once every 30 runs after a few hours
- The scientists spent 2 months in the period of 18 months, and then gave up on debugging it

**ParaDis**
- The bug intermittently manifested at 100 to 200 iteration
- The scientists gave up debugging by themselves

and more ...
How MPI introduces non-determinism?

- It’s typically due to communication with MPI_ANY_SOURCE
- In non-deterministic applications, each MPI rank doesn’t know which other MPI ranks will send message and when

```c
Non-deterministic code w/ MPI_ANY_SOURCE

MPI_Irecv(..., MPI_ANY_SOURCE, ...);
while(1) {
    MPI_Test(flag);
    if (flag) {
        <computation>
        MPI_Irecv(..., MPI_ANY_SOURCE, ...);
    }
}
```
CORAL benchmark: MCB (Monte carlo benchmark)

- Use of MPI_ANY_SOURCE is not only source of non-determinism
  - MPI_Waitany/Waitsome/Testany/Testsome also introduce non-determinism

Example: Communications with neighbors

Non-deterministic code w/o MPI_ANY_SOURCE

```c
MPI_Irecv(..., north_rank, ..., reqs[0]);
MPI_Irecv(..., south_rank, ..., reqs[1]);
MPI_Irecv(..., west_rank, ..., reqs[2]);
MPI_Irecv(..., east_rank, ..., reqs[3]);
while(1) {
    MPI_Testsome(..., reqs, ..., flag, status);
    if (flag) {
        ...
        <computation>
        for (...) MPI_Irecv(..., status.MPI_SOURCE, ...);
    }
}
```
ReMPI deterministically reproduce order of message receives

- ReMPI is an MPI record-and-replay tool
  - Record an order of MPI message receives
  - Replay the exactly same order of MPI message receives
- Even if a bug manifests in a particular order of message receives, ReMPI can consistently reproduce the target bug
- ReMPI works with other existing debugging tools
  - STAT
  - Parallel debuggers (e.g., Totalview, DDT)

SC15: Kento Sato et al., "Clock delta compression for scalable order-replay of non-deterministic parallel applications"
MCB with/without ReMPI

- Performance metric: How many particles are tracked per second

- ReMPI becomes scalable by recording to local memory/storage
  - Each rank independently writes record → No communication across MPI ranks

Recording location: Local SSDs
ReMPI case study: ParaDiS

- ParaDiS
  - non-deterministically crashed after 100 to 200 iterations
- ReMPI reproduced the bug at the exactly same iteration
- ReMPI is interoperable with parallel debuggers and makes debugging non-deterministic bug easier
  - We recorded a buggy behavior in record mode
  - We diagnosed with TotalView under replay mode
ReMPI is also useful for “Testing”

- “Testing” is also important for maintaining software quality
- However, non-deterministic MPI applications present significant challenges to testing
  - The non-determinism can produce different results from run to run by nature
  - It’s difficult to reason the different numerical results are due to MPI non-determinism or software bug
- Using ReMPI, computational scientists can easily reproduce MPI behaviors, which facilitate testing
MPI is not only source of non-determinism

- Applications have been going towards hybrid programming model
  - E.g.) MPI + OpenMP

- OpenMP code adds another level of non-determinism
  - Reduction, critical section or data racy access
  - OpenMP non-determinism affects MPI function call behaviors
  - Need to record both MPI and OpenMP events

Providing record-and-replay for MPI+OpenMP application is our future work.
Unintended message races in MPI

- Many applications are written as a series of communication and computation routines executed by all processes (i.e., data parallel, SPMD)
- Developers must make sure all communication routines are “isolated”
- Example (Routine A and Routine B)
  - Different MPI_TAG or synchronization (e.g. MPI_Barrier) between the two routines
- If not isolated, message race bugs potentially occur
  - E.g.) A message sent in Routine B is received in Routine A
- Unintended message races are non-deterministic and infrequently occur
NINJA (Noise Injection Agent Tool) exposes message race bugs by injecting noise.

NINJA detects suspicious communication routines:
- Communication routine using the same MPI_TAG without synchronization.

NINJA injects a delay to MPI messages in order to enforce message races.

NINJA can test if the application has unintended message races.

PPoPP2017: Kento Sato et al., Noise Injection Techniques for Reproducing Subtle and Unintended Message Races
NINJA cast study: Diablo/Hypre-2.10.1 (in ParaSail module)

- Unintended message races in Hypre
- Prior to NINJA, the bug does not manifest itself in Hypre test code
- NINJA consistently exposed message races in the test code

- The bug manifested in particular machines
- It hung only once every 30 runs after a few hours
- The scientists spent 2 months in the period of 18 months, and then gave up on debugging it
Summary

- Non-determinism in MPI applications costs significant time for debugging and testing

- ReMPI and NINJA facilitate debugging/testing non-deterministic MPI applications
  - ReMPI is MPI record-and-replay for reproducing particular errors
  - NINJA is an noise injection tool for exposing message-race bugs

- We will extend our tools for supporting MPI+OpenMP applications in future
PRUNERS ReMPI
OR https://github.com/PRUNERS/ReMPI

PRUNERS NINJA
OR https://github.com/PRUNERS/NINJA