ReMPI

Scalable MPI Record + Replay

Ignacio Laguna, Harshitha Menon
Lawrence Livermore National Laboratory

Michael Bentley, Ian Briggs, Pavel Panchekha, Ganesh Gopalakrishnan
University of Utah

Hui Guo, Cindy Rubio González
University of California at Davis

Michael O. Lam
James Madison University

Lawrence Livermore National Laboratory

THE UNIVERSITY OF UTAH®

UCDAVIS
UNIVERSITY OF CALIFORNIA

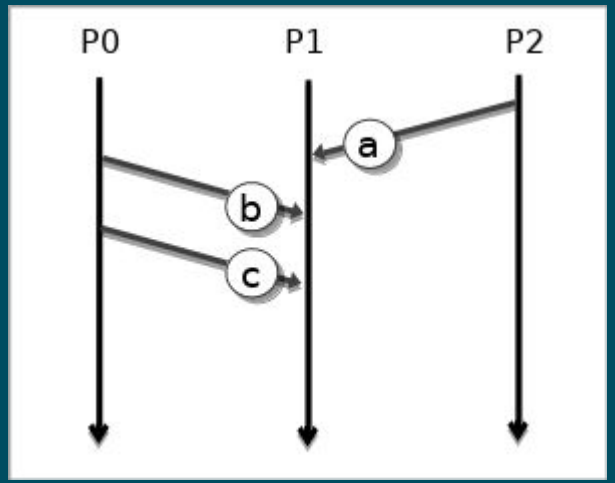
JMU
JAMES MADISON UNIVERSITY

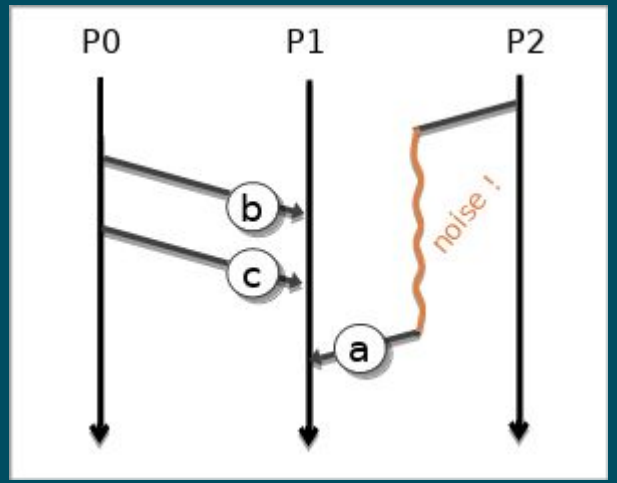
http://fpanalysistools.org/

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-PRES-780623).

MPI Non-Determinism

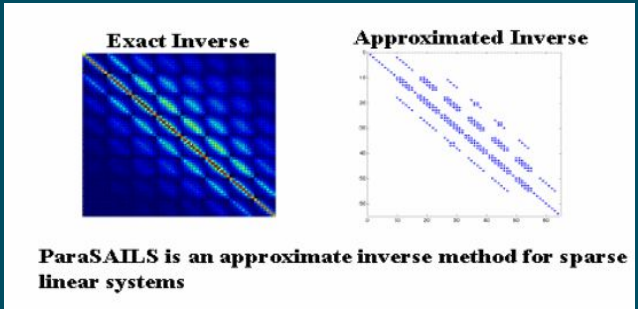
- MPI: Message Passing Interface
- Messages usually sent over a network
- Orderings may be random and could change program behavior





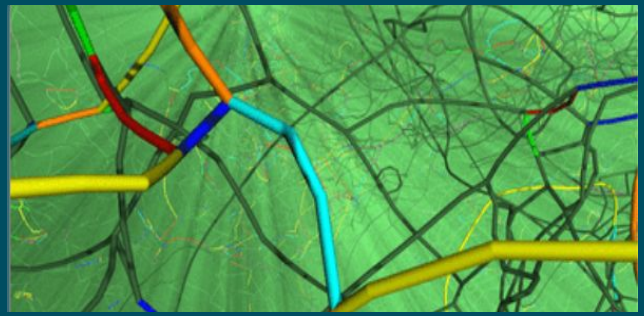
Examples

Diablo with Hypre



- Hang after many hours
- 1 in 30 runs hang
- 2 months debugging only to give up

ParaDis



- Crash between iteration 100 and 200
- Gave up debugging

Causes of MPI Non-Determinism

MPI_ANY_SOURCE

- Receives from any sender
- Can allow different orderings

```
1 MPI_Irecv(..., MPI_ANY_SOURCE, ...);
2 while (true) {
3   MPI_Test(flag);
4   if (flag) {
5     // computations...
6     MPI_Irecv(..., MPI_ANY_SOURCE, ...);
7   }
8 }
```

MPI_Testsome/MPI_Waitsome MPI_Testany/MPI_Waitany

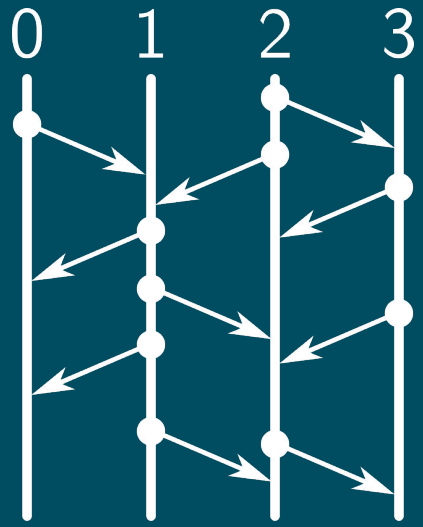
- Progress from any queued receive
- Can allow different orderings

```
 1 MPI_Irecv(..., north_rank, ..., reqs[0]);
 2 MPI_Irecv(..., south_rank, ..., reqs[1]);
 3 MPI_Irecv(..., west_rank, ..., reqs[2]);
 4 MPI_Irecv(..., east_rank, ..., reqs[3]);
 5 while (true) {
 6   MPI_Testsome(..., &reqs, &count, ..., &status);
 7   if (count > 0) {
 8     // computations...
 9     for (...) MPI_Irecv(..., status[i].MPI_SOURCE, ...);
10   }
11 }
```

MPI Record + Replay - Naive Approach

For each process record each **Send**, **Receive**, **Test**, and **Wait**

- Function type
- ID of Sender
- ID of Receiver
- Unique message ID
- Result of test
- Result of wait



Scales poorly - 24 hours of a Monte-Carlo simulation used **10GB per node**!

ReMPI

Version 1.1.0

Written by Kento Sato

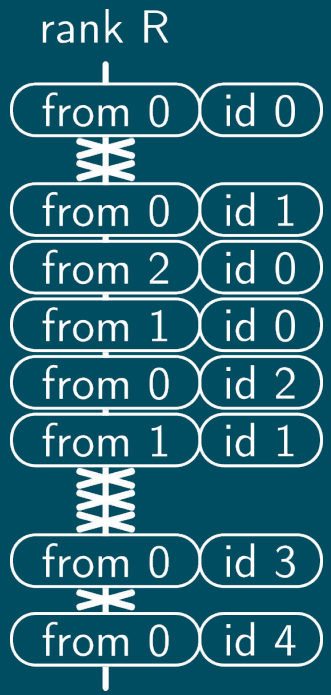
(kento.sato@riken.jp)

ReMPI Design Goals

1. Correct MPI record + replay
2. Low runtime overhead
3. Memory and file size efficiency
4. Easy to use

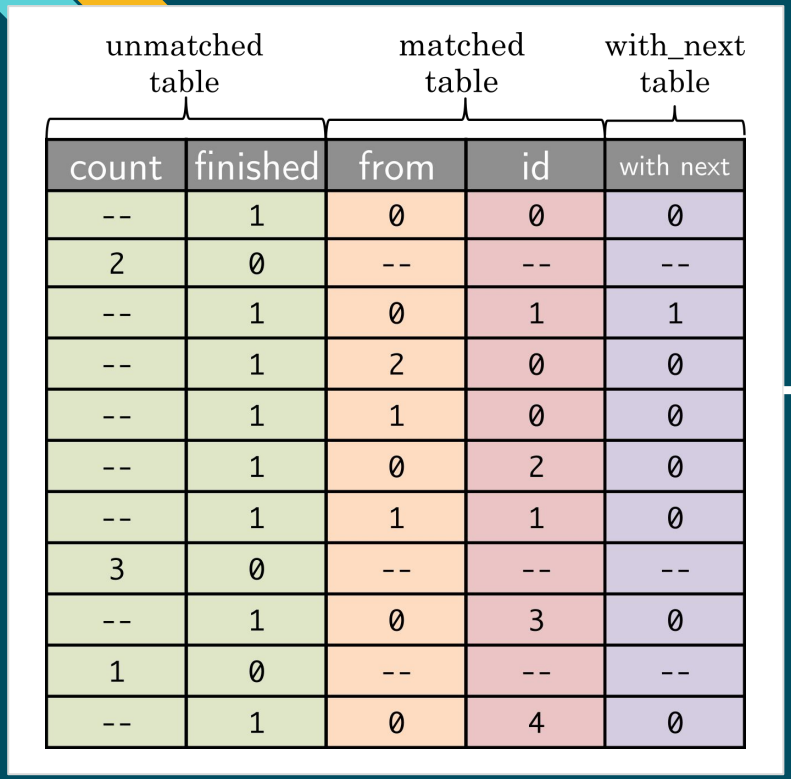
What ReMPI Captures

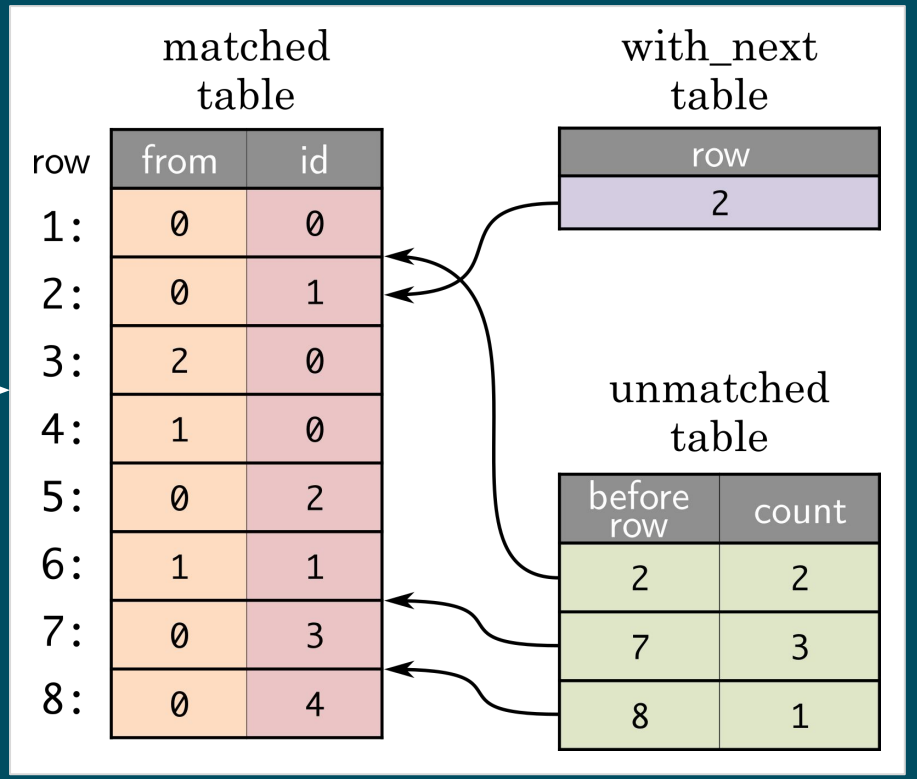
- Function type
- ID of Sender
- ID of Receiver
- Unique message ID
- Result of test
- Result of wait



unmatched table		matched table		with_next table
count	finished	from	id	with next
--	1	0	0	0
2	0	--	--	--
--	1	0	1	1
--	1	2	0	0
--	1	1	0	0
--	1	0	2	0
--	1	1	1	0
3	0	--	--	--
--	1	0	3	0
1	0	--	--	--
--	1	0	4	0

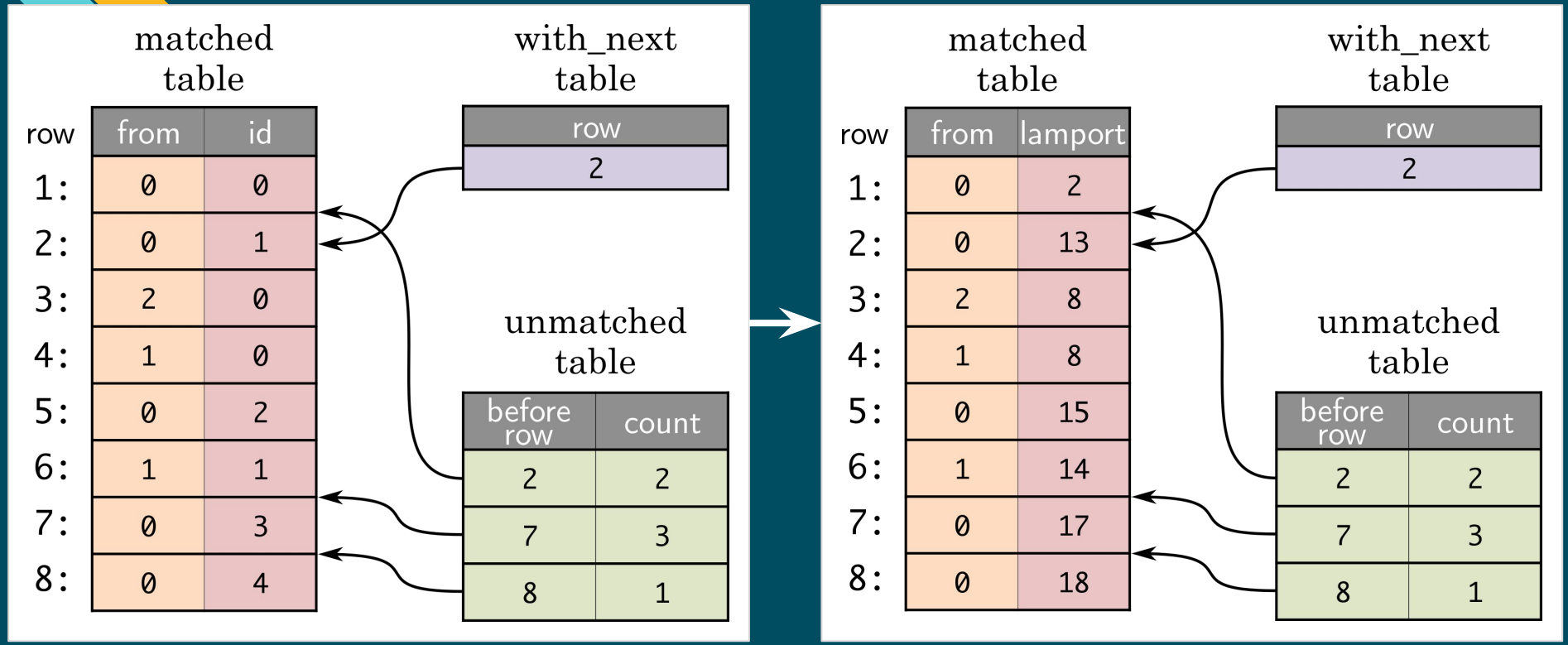
Redundancy Elimination





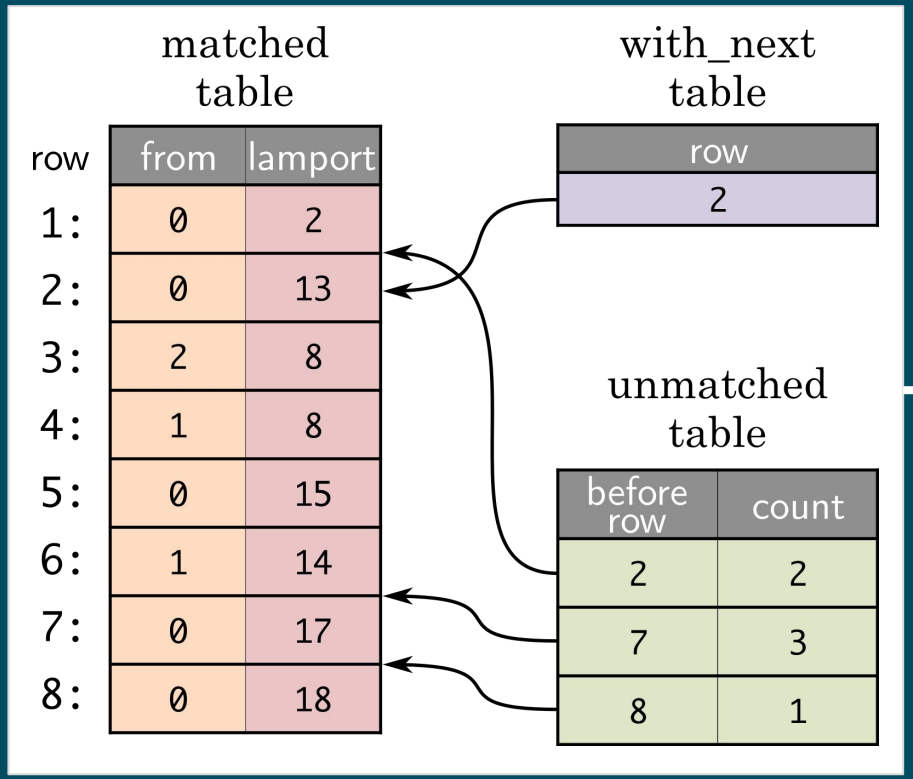
55 values → 23 values

Lamport Clocks



23 values → 23 values

Clock Delta Compression (CDC)



row	delay
2	+2
3	+1
6	−1

row
2

before row	count
2	2
7	3
8	1

http://fpanalysistools.org/

23 values → 13 values

Linear Predictive Encoding

row	delay
2	+2
3	+1
6	−1

row
2

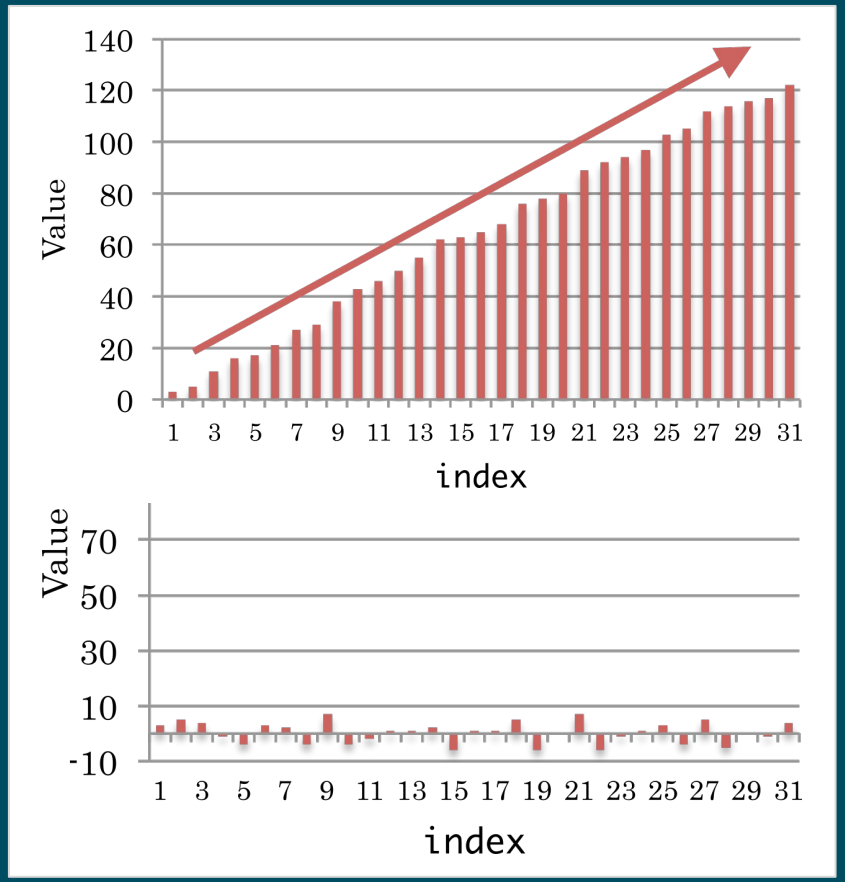
before row	count
2	2
7	3
8	1

delta p.	delay
+2	+2
+1	+1
+2	−1

delta predict.
+2

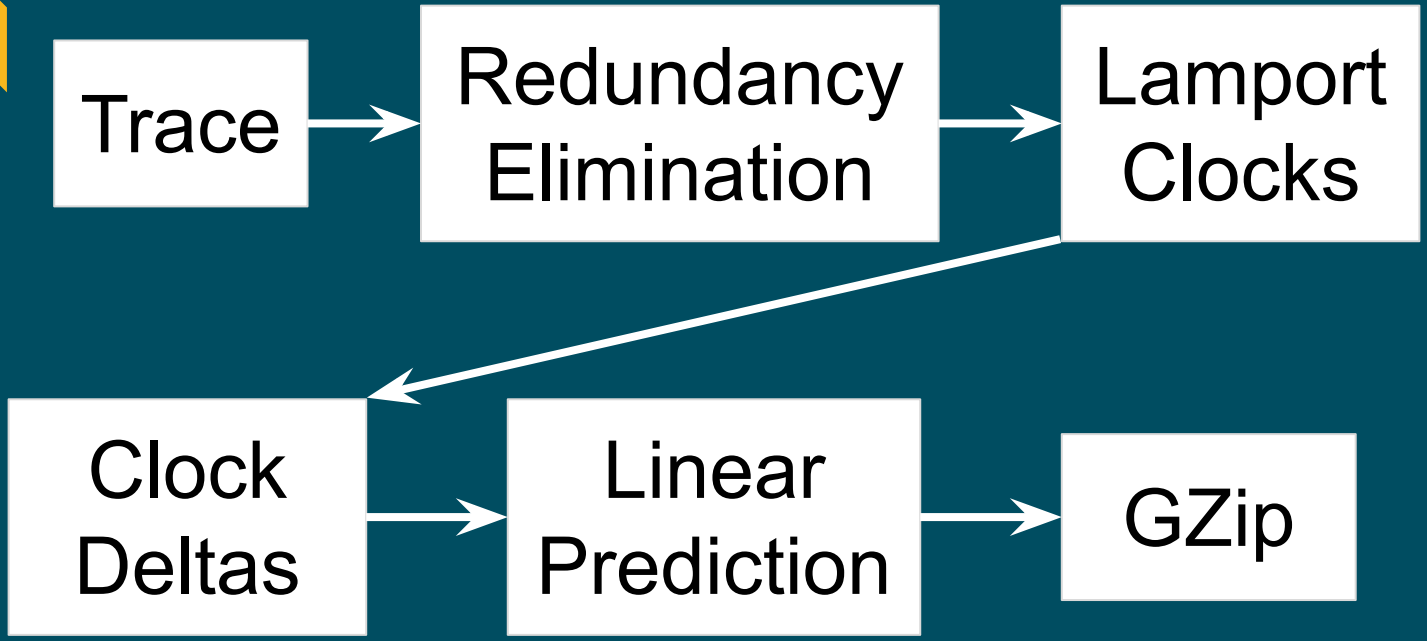
before delta	count
+2	2
+5	3
−4	1

13 values



http://fpanalysistools.org/

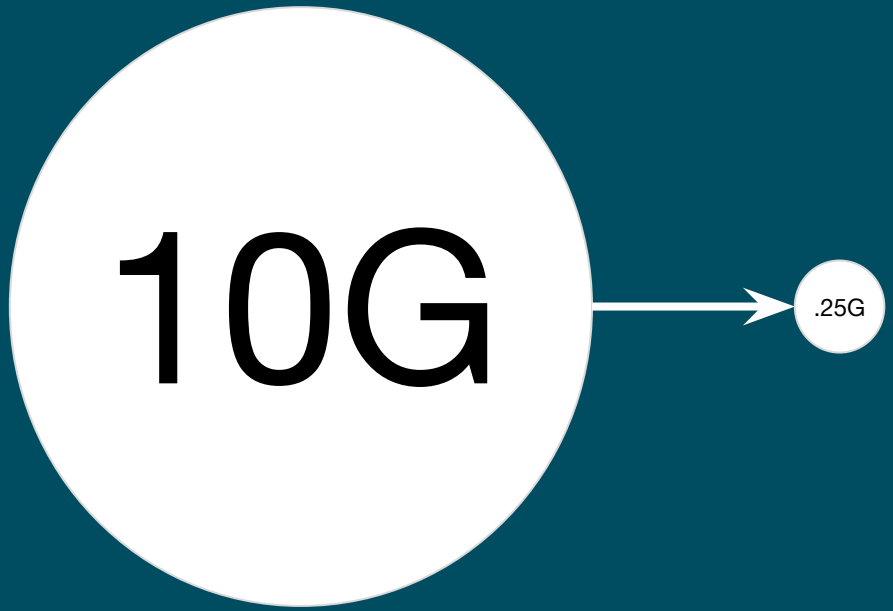
Total Pipeline



13 values

Effectiveness

40x Compression



20 %
Overhead
vs. Naive

Examples

Exercise 1 - Look at the code

```
Module-ReMPI $ cd exercise-1
```

Let’s look at the simple example MPI application

`example.c`

```
exercise-1 $ vim example.c
```

or

```
exercise-1 $ pygmentize example.c | cat -n
```

or whatever...

Exercise 1 - Look at the code

example.c

```
 9 int main(int argc, char *argv[]) {
10-20   [...]
21   for (dest = 0; dest < size; dest++) {
22
23     // each process takes a turn being the receiver
24     if (my_rank == dest) {
25       fprintf(stderr, "----\n");
26       for (i = 0; i < size-1; i++) {
27         MPI_Recv(&buf, 1, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &status);
28         fprintf(stderr, "Rank %d: MPI_Recv from Rank %d\n",
29                 my_rank, status.MPI_SOURCE);
30       }
31
32     // all other processes send
33     } else {
34       // random sleep to induce random behavior
35       usleep(rand() % 10 * 10000);
36
37       MPI_Send(&buf, 1, MPI_INT, dest, 0, MPI_COMM_WORLD);
38     }
39
40     // wait for all messages to be delivered
41     MPI_Barrier(MPI_COMM_WORLD);
42   }
```

Exercise 1 - `./step-01.sh`

```
exercise-1 $ mpicc example.c
```

Compile the example

- ReMPI is not involved with compilation

Exercise 1 - `./step-02.sh`

Run the example many times without ReMPI.

Convince yourself it changes from run to run.

```
 exercise-1 $ mpirun -n 4 ./a.out
----
Rank 0: MPI_Recv from Rank 3
Rank 0: MPI_Recv from Rank 1
Rank 0: MPI_Recv from Rank 2
----
Rank 1: MPI_Recv from Rank 2
Rank 1: MPI_Recv from Rank 3
Rank 1: MPI_Recv from Rank 0
----
Rank 2: MPI_Recv from Rank 3
Rank 2: MPI_Recv from Rank 0
Rank 2: MPI_Recv from Rank 1
----
Rank 3: MPI_Recv from Rank 2
Rank 3: MPI_Recv from Rank 0
Rank 3: MPI_Recv from Rank 1
```

Exercise 1 - `./step-03.sh`

Run ReMPI record manually

```
 exercise-1 $ REMPI_MODE=0 \
> LD_PRELOAD=/usr/local/lib/librempi.so \
> mpirun -n 4 ./a.out
REMPI::eaec2a97ea3c:  0:  ========== ReMPI Configuration ==========
REMPI::eaec2a97ea3c:  0:        REMPI_MODE:  0
REMPI::eaec2a97ea3c:  0:         REMPI_DIR:  .
REMPI::eaec2a97ea3c:  0:      REMPI_ENCODE:  0
REMPI::eaec2a97ea3c:  0:        REMPI_GZIP:  0
REMPI::eaec2a97ea3c:  0:     REMPI_TEST_ID:  0
REMPI::eaec2a97ea3c:  0:         REMPI_MAX:  131072
REMPI::eaec2a97ea3c:  0:  ========================================
[...]
REMPI::eaec2a97ea3c:  0:  Global validation code: 1732970486
```

- Uses LD_PRELOAD and PMPI
- Options are with environment variables
- Works with any MPI library

Exercise 1 - `./step-04.sh`

Run ReMPI record conveniently

```
 exercise-1 $ rempi record mpirun -n 4 ./a.out
REMPI::eaec2a97ea3c:  0:  ========== ReMPI Configuration ==========
REMPI::eaec2a97ea3c:  0:        REMPI_MODE:  0
REMPI::eaec2a97ea3c:  0:         REMPI_DIR:  .
REMPI::eaec2a97ea3c:  0:      REMPI_ENCODE:  0
REMPI::eaec2a97ea3c:  0:        REMPI_GZIP:  0
REMPI::eaec2a97ea3c:  0:     REMPI_TEST_ID:  0
REMPI::eaec2a97ea3c:  0:         REMPI_MAX:  131072
REMPI::eaec2a97ea3c:  0:  ========================================
[...]
REMPI::eaec2a97ea3c:  0:  Global validation code: 1732970486
```

- Convenience script “rempi”
- Sets LD_PRELOAD and REMPI_MODE
- Running many times still has different results

Exercise 1

See the recorded traces

```
 exercise-1 $ ls -l *.rempi
-rw-r--r-- 1 rempi sudo 296 Nov  6 07:19 rank_0.rempi
-rw-r--r-- 1 rempi sudo 296 Nov  6 07:19 rank_1.rempi
-rw-r--r-- 1 rempi sudo 296 Nov  6 07:19 rank_2.rempi
-rw-r--r-- 1 rempi sudo 296 Nov  6 07:19 rank_3.rempi
```

- Traces are put into the current directory by default
- Each process (i.e. rank) makes its own trace
- Binary files - small in size

Exercise 1 - `./step-05.sh`

Run ReMPI replay manually

```
 exercise-1 $ REMPI_MODE=1 \
> LD_PRELOAD=/usr/local/lib/librempi.so \
> mpirun -n 4 ./a.out
REMPI::eaec2a97ea3c:  0:  ========== ReMPI Configuration ==========
REMPI::eaec2a97ea3c:  0:        REMPI_MODE:  1
REMPI::eaec2a97ea3c:  0:         REMPI_DIR:  .
REMPI::eaec2a97ea3c:  0:      REMPI_ENCODE:  0
REMPI::eaec2a97ea3c:  0:        REMPI_GZIP:  0
REMPI::eaec2a97ea3c:  0:     REMPI_TEST_ID:  0
REMPI::eaec2a97ea3c:  0:         REMPI_MAX:  131072
REMPI::eaec2a97ea3c:  0:  ========================================
[...]
REMPI::eaec2a97ea3c:  0:  Global validation code: 1732970486
```

- Only difference: REMPI_MODE=1
- Running many times gives *the same result!*

Exercise 1 - `./step-06.sh`

Run ReMPI replay conveniently

```
 exercise-1 $ rempi replay mpirun -n 4 ./a.out
REMPI::eaec2a97ea3c:  0:  ========== ReMPI Configuration ==========
REMPI::eaec2a97ea3c:  0:        REMPI_MODE:  1
REMPI::eaec2a97ea3c:  0:         REMPI_DIR:  .
REMPI::eaec2a97ea3c:  0:      REMPI_ENCODE:  0
REMPI::eaec2a97ea3c:  0:        REMPI_GZIP:  0
REMPI::eaec2a97ea3c:  0:     REMPI_TEST_ID:  0
REMPI::eaec2a97ea3c:  0:         REMPI_MAX:  131072
REMPI::eaec2a97ea3c:  0:  ========================================
[...]
REMPI::eaec2a97ea3c:  0:  Global validation code: 1732970486
```

- Convenience script “rempi” again
- Sets LD_PRELOAD and REMPI_MODE

Exercise 1 - `./step-07.sh`

Try replay with different process count

```
 exercise-1 $ rempi replay mpirun -n 5 ./a.out
[...]
REMPI: ** ERROR **:eaec2a97ea3c:  4:  Record file open failed: ./rank_4.rempi
(rempi_encoder.cpp:open_record_file:226)
a.out: rempi_err.cpp:95: void rempi_assert(int): Assertion `b' failed.
Rank 0: MPI_Recv from Rank 1
Rank 0: MPI_Recv from Rank 2
Rank 0: MPI_Recv from Rank 3
REMPI:ALERT:eaec2a97ea3c:  0:  MPI_Recv/Irecv should not be called according to record: 2
(MPI_Recv/Irecv: 1, Matching function: 2, Probing function: 3)
(rempi_recorder.cpp:replay_irecv:370)
a.out: rempi_err.cpp:95: void rempi_assert(int): Assertion `b' failed.

[...]
```

Fails fast and hard when used wrong

Exercise 1 - `./step-08.sh`

Try replay with different process count

```
 exercise-1 $ rempi replay mpirun -n 3 ./a.out
[...]
REMPI:ALERT:eaec2a97ea3c:  0:  A matching function should not be called according to
record: 1
  (MPI_Recv/Irecv: 1, Matching function: 2, Probing function: 3)
(rempi_recorder.cpp:replay_mf_input:945)
a.out: rempi_err.cpp:95: void rempi_assert(int): Assertion `b' failed.

[...]
```

Fails fast and hard when used wrong

ReMPI Options

Options are printed at the top of the output

```
REMPI::eaec2a97ea3c:  0:  ========== ReMPI Configuration ==========
REMPI::eaec2a97ea3c:  0:        REMPI_MODE:  1
REMPI::eaec2a97ea3c:  0:         REMPI_DIR:  .
REMPI::eaec2a97ea3c:  0:      REMPI_ENCODE:  0
REMPI::eaec2a97ea3c:  0:        REMPI_GZIP:  0
REMPI::eaec2a97ea3c:  0:     REMPI_TEST_ID:  0
REMPI::eaec2a97ea3c:  0:         REMPI_MAX:  131072
REMPI::eaec2a97ea3c:  0:  ========================================
[...]
```

I will show:

- REMPI_DIR
- REMPI_GZIP

Exercise 1 - `./step-09.sh`

Record to a given directory using environment variable

```
 exercise-1 $ export REMPI_DIR=./rempi-races
 exercise-1 $ rempi record mpirun -n 4 ./a.out
[...]
 exercise-1 $ ls -l ./rempi-races
total 16
-rw-r--r-- 1 rempi sudo 264 Nov  6 15:21 rank_0.rempi
-rw-r--r-- 1 rempi sudo 296 Nov  6 15:21 rank_1.rempi
-rw-r--r-- 1 rempi sudo 264 Nov  6 15:21 rank_2.rempi
-rw-r--r-- 1 rempi sudo 296 Nov  6 15:21 rank_3.rempi
```

You can set the environment variable once and work

Exercise 1 - `./step-10.sh`

Record to a given directory using argument

```
 exercise-1 $ rempi record REMPI_DIR=./rempi-races mpirun -n 4 ./a.out
[...]
 exercise-1 $ ls -l ./rempi-races
total 16
-rw-r--r-- 1 rempi sudo 264 Nov  6 15:21 rank_0.rempi
-rw-r--r-- 1 rempi sudo 296 Nov  6 15:21 rank_1.rempi
-rw-r--r-- 1 rempi sudo 264 Nov  6 15:21 rank_2.rempi
-rw-r--r-- 1 rempi sudo 296 Nov  6 15:21 rank_3.rempi
```

You can give it as an argument each time instead

Exercise 1 - `./step-11.sh`

Replay from a given directory using argument

```
 exercise-1 $ rempi replay \
> REMPI_DIR=./rempi-races \
> mpirun -n 4 ./a.out
[...]
```

If you do not have the REMPI_DIR environment variable set,
then you need to specify it at replay too.

Exercise 1 - `./step-12.sh`

Record a large run with GZip

```
 exercise-1 $ rempi record \
> REMPI_DIR=./rempi-gzip \
> REMPI_GZIP=1 \
> mpirun -n 20 ./a.out
[...]
 exercise-1 $ ls -l ./rempi-gzip
total 80
-rw-r--r-- 1 rempi sudo 174 Nov  6 16:14 rank_0.rempi
-rw-r--r-- 1 rempi sudo 164 Nov  6 16:14 rank_1.rempi
-rw-r--r-- 1 rempi sudo 175 Nov  6 16:14 rank_10.rempi
-rw-r--r-- 1 rempi sudo 175 Nov  6 16:14 rank_11.rempi
[...]
```

The compressed traces look small.

Let’s see how big without gzip

Exercise 1 - `./step-13.sh`

Record a large run without GZip for comparison

```
 exercise-1 $ rempi record \
> REMPI_DIR=./rempi-no-gzip \
> REMPI_GZIP=0 \
> mpirun -n 20 ./a.out
[...]
 exercise-1 $ ls -l ./rempi-no-gzip
total 80
-rw-r--r-- 1 rempi sudo 1832 Nov  6 16:19 rank_0.rempi
-rw-r--r-- 1 rempi sudo 1832 Nov  6 16:19 rank_1.rempi
-rw-r--r-- 1 rempi sudo 1832 Nov  6 16:19 rank_10.rempi
-rw-r--r-- 1 rempi sudo 1832 Nov  6 16:19 rank_11.rempi
[...]
```

The uncompressed traces are about 11x bigger.

Exercise 1 - `./step-14.sh`

Replay a GZip run

```
 exercise-1 $ rempi replay \
> REMPI_DIR=./rempi-gzip \
> REMPI_GZIP=1 \
> mpirun -n 20 ./a.out
[...]
```

You must specify the same REMPI_GZIP setting to replay
I suggest you set it in your environment variables

ReMPI

Thank You!

Questions?

pruners.github.io/rempi