18th European Workshop on AD

Checkpointing on Adjoint MPI Programs

INRIA Sophia-Antipolis, France

Presented by
Ala Taftaf

Supervised by:
Laurent Hascoët
Contents

1 Introduction

2 Checkpointing Adjoint MPI Programs: “Memo technique”

3 Re-sending

4 Future work
Checkpointing in the Store All context

- Select a piece of code "checkpointed piece" and not store its intermediate values.
- Store only the values needed to reexecute the checkpointed piece later (a "snapshot");
- The checkpointed piece is executed again, storing the intermediate values.

\[
\begin{align*}
\vec{U} & \rightarrow \vec{C} & \vec{D} & \rightarrow & \vec{C} & \vec{D} & \rightarrow & \vec{C} & \vec{D} \\
\vec{C} & \rightarrow & \vec{D} & \rightarrow & \vec{C} & \rightarrow & \vec{C} & \rightarrow & \vec{U}
\end{align*}
\]
The checkpointed piece of source code may correspond at run time to several checkpointed intervals of execution "checkpoints".

At run time, the nested structure of checkpoints form a tree.
Checkpointing adjoint MPI programs (point-to-point communications)

- Communications restrict application of Checkpointing
- Popular approach: checkpointing only occurs at a level that **encompasses** the level where communication takes place. In particular:
  - Both ends of each communication must be checkpointed in the same way.
  - Non blocking routines (e.g. isend) and their waits must be checkpointed together.
Popular MPI checkpointing is not general

If only one end of a point-to-point communication is checkpointed, the resulting code fails.

```
Process1:
mpi_send(a)

Process2:
mpi_recv(b)
```

```
Process1:
mpi_send(a)

Process2:
mpi_send(a)
mpi_recv(x)
```

```
Process1:
mpi_recv(y)

Process2:
mpi_recv(y)
```
Another problem: nonblocking communications

If the non blocking routine doesn’t belong to the same checkpoint as its wait, the resulting code fails

⇒ Need to lift this restriction
Checkpointing Adjoint MPI Programs: “Memo technique”

1. Introduction

2. Checkpointing Adjoint MPI Programs: “Memo technique”

3. Re-sending

4. Future work
Checkpointing Adjoint MPI Programs: Memo technique

- `mpi_recv` log their received values. Same thing for the `mpi_wait` of a `mpi_irecv`.
- Repeated `mpi_send` are disabled. Same thing for `mpi_isend`, `mpi_irecv` and `mpi_wait` of a `mpi_isend`.
- Repeated `mpi_recv` are replaced by a read of the logged value. Same thing for the `mpi_wait` of `mpi_irecv`.

```
Process1:
mpi_send(a)
Process2:
mpi_recv(b)
no_op
```
```
Process1:
mpi_recv(y); log(y)
Process2:
mpi_send(x)
retrieve(y)
```
What if nested checkpoints?

Process1:
- recv(b,0) → mpi_recv(b); log(b)
- recv(b,1) → retrieve(b)
- recv(b,2) → retrieve(b)

Process2:
- send(a,0) → mpi_send(a)
- send(a,1) → no_op
Discussion on the memo technique

The memo technique:

- Changes the behavior of communication calls
- Requires adaptation of the checkpoint mechanism: the logged values (conceptually a part of the snapshot) do not follow the stack order.
- Has no specific conditions in the choice of the checkpoints.
- Lets each process be checkpointed independently from other process.
Memory issues

- Logging values uses memory
- Messages are often large
- Non-stack structure prevents memory reuse

⇒ The memo technique is general, but memory-costly
Re-sending

1. Introduction
2. Checkpointing Adjoint MPI Programs: “Memo technique”
3. Re-sending
4. Future work
Re-sending

- Repeat communications whenever possible ⇒ this reduces logging size.
- The 2 ends of a repeated communication must be at the same checkpointing level.
When is resending possible?

- If the “re-send” communication is non-blocking, its wait must belong to the same checkpoint level.
- Defining checkpoints as “paired” the checkpoints that contain the 2 ends of a “resend” communication, one checkpoint in a process cannot be paired with two checkpoints in another.
Future work

1. Introduction

2. Checkpointing Adjoint MPI Programs: “Memo technique”

3. Re-sending

4. Future work
Future work

- Proof of correction
- Implementation in Tapenade and AMPI.
- Experiments on real codes
Acknowledgements

This work has been conducted within the **About Flow** project on “Adjoint-based optimisation of industrial and unsteady flows”.

http://aboutflow.sems.qmul.ac.uk

About Flow has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under Grant Agreement No. 317006.