

The data-flow equations of Checkpointing

Laurent Hascoët, Benjamin Dauvergne
<http://www-sop.inria.fr/tropics>

2nd Euro AD workshop, Cranfield, Nov. 2005

- Reverse AD by program transformation
(\Rightarrow opportunity for data-flow analysis: activity, . . .)
- Store-All approach (\Rightarrow needs optimized taping, TBR)
- Nested Checkpointing
(\Rightarrow repeated executions, Snapshots)

For the Data Flow Equations we use in AD tools, we need a **formal justification** and a guarantee of “**optimality**”.

Checkpointing questions

When **no** checkpointing is done:

- Unique optimal answer for activity, adj-liveness, TBR
- Data Flow Equations derived formally
- No retroaction between analyses

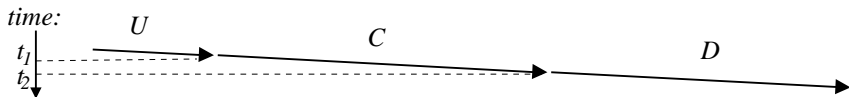
But when checkpointing **is** present:

- Still no problem for activity and adj-liveness
- Examples show many optimal answers for TBR/Snapshots
- Retroaction between TBR and Snapshot

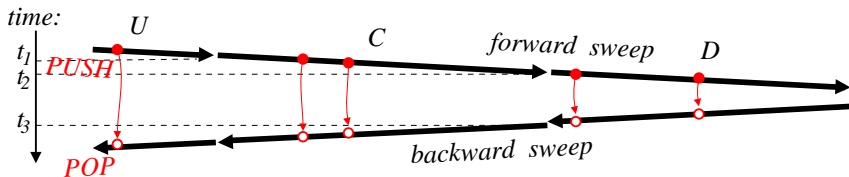
⇒ Our goal is to **characterize** all possible “optimal” strategies for TBR/Snapshot, and then experiment some of them on real applications.

Reverse AD without Checkpointing

- Original program $U; C; D$:

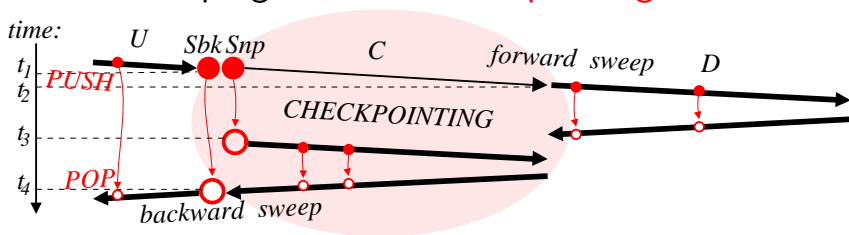


- Reverse diff program, no checkpointing:

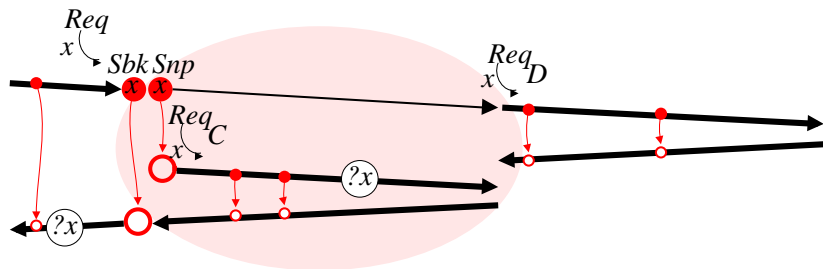


Checkpointing tactique, Snapshots, TBR

- Reverse diff program, with Checkpointing on C:



The retroaction problem



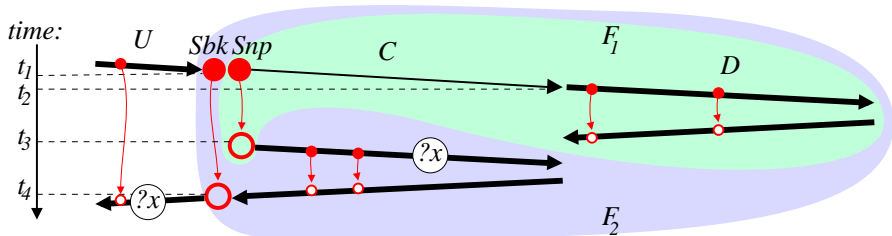
Variable x may be needed

- either in \overleftarrow{U} (TBR)
- or in \overline{C} (Checkpointing)

In either case, there are **many** ways to preserve x .

⇒ Need to be systematic

Developping the **out** sets



$$\mathbf{out}(F_1) = (\mathbf{out}(C) \cup (\mathbf{out}(\overline{D}) \setminus Req_D)) \setminus Snp$$

$$\mathbf{out}(F_2) = ((\mathbf{out}(C) \cup (\mathbf{out}(\overline{D}) \setminus Req_D)) \setminus Snp \cup (\mathbf{out}(\overline{C}) \setminus Req_C)) \setminus Sbk$$

Equations for the minimal solutions

$$Sbk \supseteq ((\mathbf{out}(C) \cup (\mathbf{out}(\bar{D}) \setminus Req_D)) \setminus Snp) \cup (\mathbf{out}(\bar{C}) \setminus Req_C) \cap Req$$

$$Snp \supseteq (\mathbf{out}(C) \cup (\mathbf{out}(\bar{D}) \setminus Req_D)) \cap (\mathbf{use}(\bar{C}) \cup (Req \setminus Sbk))$$

$$Req_D \supseteq (\mathbf{out}(\bar{D}) \setminus Snp) \cap (\mathbf{use}(\bar{C}) \cup (Req \setminus Sbk))$$

$$Req_C \supseteq (\mathbf{out}(\bar{C}) \setminus Sbk) \cap Req$$

- Retroaction is now apparent
 - Hand resolution error-prone
- ⇒ use a symbolic computation tool

The minimal solutions

Define:

$$Snp_0 = \mathbf{out}(C) \cap (\mathbf{use}(\overline{C}) \cup (Req \setminus \mathbf{out}(\overline{C})))$$

$$Opt_1 = Req \cap \mathbf{out}(\overline{C}) \cap \mathbf{use}(\overline{C})$$

$$Opt_2 = Req \cap \mathbf{out}(\overline{C}) \setminus \mathbf{use}(\overline{C})$$

$$Opt_3 = \mathbf{out}(\overline{D}) \cap (\mathbf{use}(\overline{C}) \cup Req) \setminus \mathbf{out}(C)$$

Every minimal solution is of the form:

$$Sbk = Opt_1^+ \cup Opt_2^+$$

$$Snp = Snp_0 \cup Opt_2^- \cup Opt_3^+$$

$$Req_D = Opt_3^-$$

$$Req_C = Opt_1^- \cup Opt_2^-$$

“Eager Snapshots”

Take $Opt_1^+ = Opt_1$, $Opt_2^+ = Opt_2$, and $Opt_3^+ = Opt_3 \Rightarrow$

$$Sbk = Req \cap \mathbf{out}(\overline{C})$$

$$Snp = (Req \cap \mathbf{out}(\overline{D}) \setminus \mathbf{out}(C))$$

$$\cup (Req \cap \mathbf{out}(C) \setminus \mathbf{out}(\overline{C}))$$

$$\cup (\mathbf{use}(\overline{C}) \cap \mathbf{out}(\overline{D})) \cup (\mathbf{use}(\overline{C}) \cap \mathbf{out}(C))$$

$$Req_D = \emptyset$$

$$Req_C = \emptyset$$

- Need $\mathbf{out}(\overline{C})$, $\mathbf{out}(\overline{D})$
- Snapshot anticipates TBR \Rightarrow rarely good...

“Lazy Snapshots”

Take $Opt_1^+ = \emptyset$, $Opt_2^+ = \emptyset$, and $Opt_3^+ = \emptyset \Rightarrow$

$$Sbk = \emptyset$$

$$Snp = \mathbf{out}(C) \cap (Req \cup \mathbf{use}(\overline{C}))$$

$$Req_D = \mathbf{out}(\overline{D}) \cap (Req \cup \mathbf{use}(\overline{C})) \setminus \mathbf{out}(C)$$

$$Req_C = \mathbf{out}(\overline{C}) \cap Req$$

- Saves are delayed until the very last moment
- No need for $\mathbf{out}(\overline{C})$, $\mathbf{out}(\overline{D})$
- Best strategy in general, except for special (contrived) cases.

Memory measurements

<i>Code</i>	<i>Domain</i>	<i>time</i>	<i>Eager</i>	<i>Lazy</i>
OPA	oceanography	780 s	480 Mb	479 Mb
STICS	agronomy	35 s	229 Mb	229 Mb
UNS2D	CFD	23 s	248 Mb	185 Mb
SAIL	agronomy	17 s	1.6 Mb	1.5 Mb
THYC	thermodynamics	12 s	33.7 Mb	18.3 Mb
LIDAR	optics	10 s	14.6 Mb	14.6 Mb
CURVE	shape optim	2.7 s	1.44 Mb	0.59 Mb
SONIC	CFD	0.2 s	3.55 Mb	2.02 Mb

Lazy snapshots never loose on real applications.
Gain less visible on long iterative programs.

Nested Checkpoints

What is the relative influence of nested checkpoints?

Optimal sets depend on $\mathbf{use}(\overline{C})$, $\mathbf{out}(\overline{C})$, $\mathbf{out}(\overline{D})$.

Does $\mathbf{out}(\overline{P})$ depend on the checkpoints inside P ?

(Maple:) \Rightarrow whatever the choice of Opt_1^+ , Opt_2^+ , Opt_3^+ ,
the value of $\mathbf{out}(\overline{C}; \overline{D})$ is the same:

$$\mathbf{out}(\overline{C}; \overline{D}) = (\mathbf{out}(\overline{C}) \cap (\mathbf{out}(\overline{D}) \cup \mathbf{out}(C)) \setminus \mathbf{use}(\overline{C})) \setminus \text{Req}$$

- Adaptive choice of Opt_1^+ , Opt_2^+ , Opt_3^+ , different for each checkpoint.
- Activation and deactivation of checkpoints based on the data-flow **use** and **out** sets.
- Measurements should look not only at tape peak, but also at tape traffic.