Extending the Binomial Checkpointing Technique for Resilience*

Andrea Walther[1] Sri Hari Krishna Narayanan[2]

1 Introduction

The usage of adjoint methods allows the computation of gradient information within a time that is only a very small multiple of the time needed to evaluate the underlying function itself. However, as soon as the considered process is nonlinear, the memory requirement to compute the adjoint information is in principle proportional to the operation count of the underlying function. Checkpointing strategies alleviate the high memory complexity using a small number of memory units (checkpoints) to store the system state at distinct times. Subsequently, the recomputation of information that is needed for the adjoint computation but not available is performed using these checkpoints in an appropriate way. Several checkpointing techniques have been developed all of which seek an acceptable compromise between memory requirement and runtime increase.

We assume that the evaluation of the function of interest has a time-step structure given by

$$x_i = F_i(x_{i-1}, u_{i-1}), \quad i = 1, \ldots, l \ , \tag{1.1}$$

for a given x_0, where $x_i \in \mathbb{R}^n$, $i = 0, \ldots, l$, denote the state of the considered system and $u_i \in \mathbb{R}^m$ the control. The operator $F_i : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ defines the time step to compute the state x_i. The process to compute x_l for a given x_0 is also called forward integration. To optimize a specific criterion or to obtain a desired state, the cost functional

$$J(x(u), u) = J(x, u)$$

measures the quality of $x(u) = (x_1, \ldots, x_l)$ and $u = (u_1, \ldots, u_l)$, where $x(u)$ depends on the control u. For applying a derivative-based optimization method, one could use an adjoint integration of the form

$$\begin{gathered} \bar{u}_l = 0 \ , \bar{x}_l \text{ given} \\ (\bar{x}_{i-1}, \bar{u}_{i-1}) = \bar{F}_i(\bar{x}_i, \bar{u}_i, x_{i-1}, u_{i-1}), \ i = l, \ldots, 1, \end{gathered} \tag{1.2}$$

where the operator $\bar{F}_i$ denotes the adjoint time step. Subsequently or concurrently to the adjoint integration, the desired derivative information $J_u(x(u), u)$ can be reconstructed from $\bar{x}$. As can be seen, the information of the forward integration (1.1) is needed for the adjoint computation (1.2). To provide this information within only a limited amount of memory, we use the binomial checkpointing approach proposed in [1, 2] as a basis to develop a checkpointing approach that can also handle a failure of the computing system. This includes a foreseen suspension, where the application should suspend itself gracefully after completing the set number of forward or adjoint time steps. However, also an unforeseen failure has to be covered, where the application is killed externally because of machine failure or expired time allocation. It was shown in [2] that a checkpointing scheme based on binomial coefficients yields for a given number of checkpoints the minimal number of time steps to be recomputed.

The ability to recover from a possible failure poses two additional challenges for the checkpointing scheme. First, the distance between two checkpoints should not be too large such that a restart of the computation is not too costly. Hence, there has to be an additional bound on the distance of two checkpoints in terms of the number of time steps that are performed before the next checkpoint is set. Second, since a failure may also occur during the adjoint computation also the adjoint state has to be checkpointed. Due to the nature of the adjoint computation, only one adjoint state is required to restart the adjoint computation.

If the failure happens during the forward integration, then the computation can just restart at the last checkpoint stored. If the failure happens after the start of the adjoint computation, then it may happen that the checkpoint distribution at the time of the failure differs from the one when the last adjoint checkpoint was written.

We will illustrate the deviation of the extended checkpointing approach from the optimal binomial checkpointing using the adjoint computation for the fol-

*This work was funded in part by a grant from DAAD Project Based Personnel Exchange Programme and by a grant from the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

[1]Universität Paderborn

[2]Argonne National Laboratory

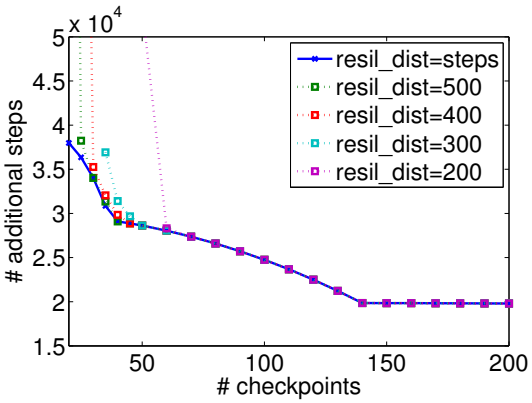

Figure 1: Additional time steps needed for adjointing 10,000 steps

lowing small academic test case

$$\begin{aligned} & \min J(x,u) \quad \text{with} && J(x,u) \equiv x_2(1), \\ \text{s.t. } & x_1'(t) = 0.5x_1(t) + u(t), && x_1(0) = 1 \\ & x_2'(t) = x_1(t)^2 + 0.5u(t)^2, && x_2(0) = 0 \\ & t \in [0,1]\ . \end{aligned}$$

Because the adjoint for this optimization problem can be derived analytically yielding

$$\begin{aligned} \lambda_1'(t) &= -0.\lambda_1(t) - 2 * x_1(t)\lambda_2(t) && \lambda_1(1) = 0 \\ \lambda_2'(t) &= 0 && \lambda_2(1) = 1 \end{aligned}$$

it is possible to verify the correctness of the adjoint computation also for the checkpointing with resilience, i.e., with the restart using the information stored in the additional files. We tested and verified the binomial checkpointing with resilience for up to 100,000 time steps and failures occurring at numerous different places. As a representative observation, Fig. 1 illustrates the additional recomputations needed as a solid line for 10,000 steps and a varying number of checkpoints. The additional recomputations needed by the binomial checkpointing approach that incorporates resilience are illustrated with dotted lines for the resilience distances of 200, 300, 400, and 500 steps. Here, one has to note that the number of checkpoints denoted with c and the resilience distance denoted with d can not be chosen completely independent from each other. Because d is the maximal number of time steps between two consecutive checkpoints, it must hold for the computation to be adjointed comprising of l time steps that

$$l \leq d \cdot c\ .$$

This bound limits the resilience distance from below for a very small number of checkpoints as illustrated also in Fig. 1. If l is close to the upper bound $d \cdot c$ plenty of recomputations have to be performed since this corresponds to the strategy of complete recomputation for a large part of the forward integration. This explains the very high number of additional time steps required for a c, d- combination where the product of both values is close to l. On the other hand, it can be seen for this example that the binomial checkpointing with resilience only interferes with the optimality of the binomial checkpointing if the number of checkpoints is less than 0.6 % of the computed intermediate states.

The checkpointing strategy implemented in [2] is in most cases only one out of a whole variety of choices that would lead to a minimal number of time step recomputations. This approach was taken because it also minimizes the number of times a checkpoint is written [2, Prop. 2]. If one wants to minimize the distance between two consecutive checkpoints, for example for resilience, a completely different strategy is used for setting the next checkpoint.

We will present a modified binomial checkpointing algorithm that supports the restart of the adjoint computation after a failure of the computing system. The modified algorithm maintains the optimality of binomial checkpointing while limiting the maximum distance between successive forward and adjoint checkpoints. The required changes were integrated in the software package revolve for binomial checkpointing and will be made available on the web site of revolve as stated at the tool list web site on www.autodiff.org. We plan to apply the resilient binomial checkpointing algorithm to compute the adjoint of the MIT General Circulation Model [3], where previously the original binomial checkpointing algorithm was used.

References

[1] A. Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, 1:35–54, 1992.

[2] A. Griewank and A. Walther. Algorithm 799: Revolve: An implementation of checkpoint for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software*, 26(1):19–45, 2000.

[3] Alistair Adcroft, Chris Hill, and John Marshall. Representation of topography by shaved cells in a height coordinate ocean model. *Monthly Weather Review*, 125(9):2293–2315, 1997.