Scientific Computing
University of Kaiserslautern-Landau

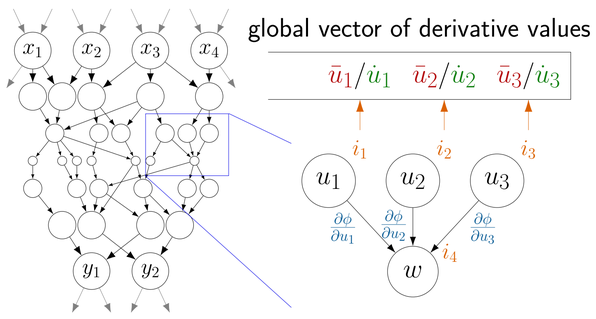
NHR@SW

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Simultaneous Preaccumulations in OpenMP-Parallel Automatic Differentiation

Johannes Blühdorn, Nicolas R. Gauger
Chair for Scientific Computing, University of Kaiserslautern-Landau (RPTU)

Automatic Differentiation (AD)



(sub)graph/program G individual statement

primal program

$w = \phi(u) \qquad y = G(x)$

reverse mode

$\bar{u}_i \mathrel{+}= \frac{\partial\phi}{\partial u_i}(u)\bar{w} \;\forall i$

$\bar{x} = \frac{\partial}{\partial x}G(x)^\mathsf{T}\bar{y}$

forward mode

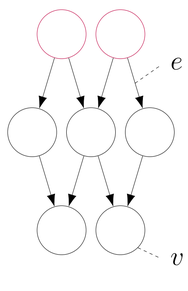
$\dot{w} = \Sigma_i \frac{\partial\phi}{\partial u_i}(u)\dot{u}_i$

$\dot{y} = \frac{\partial}{\partial x}G(x)\dot{x}$

- recorded computational graph with values as nodes, direct dependencies as edges [11]
  - edges annotated with partials, nodes annotated with identifiers (virtual addresses)
  - identifiers address into memory for adjoint values or tangent values
- compute derivatives based on graph, using the rules for the forward or reverse mode
- **preaccumulation:** precompute Jacobians of subgraphs to save memory [11]
  - apply the same techniques recursively during recording

Model for Simultaneous Preaccumulations

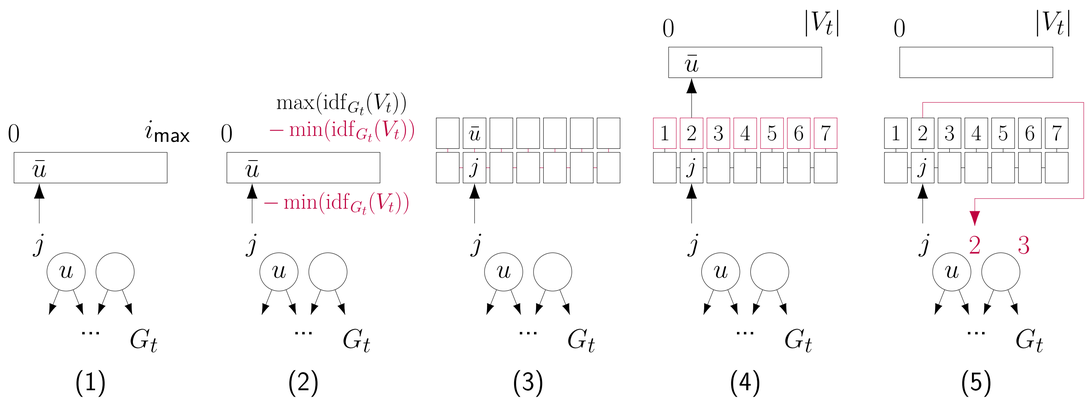
```
#pragma omp parallel
{
  // thread-local preaccumulation of subgraph G_t
}
```



- isomorphic subgraphs $G_t = (V_t, E_t)$
  - identical structure, different data/identifiers
  - input nodes may be shared among graphs
  - identifiers $\mathrm{idf}_{G_t}\colon V_t \to \mathbb{N}$
  - w.l.o.g. $|\mathrm{idf}_{G_t}(V_t)| = |V_t|$
- preaccumulated in parallel with T threads
- i_max largest identifier among all G_t and already recorded prior parts of the global graph

- simultaneous preaccs. with a shared vector of derivative values are subject to **data races**
- threads place/combine unrelated values in derivative memory for shared inputs
- use **thread-local derivative memory** instead [4]

Alternatives for Thread-Local Derivative Memory

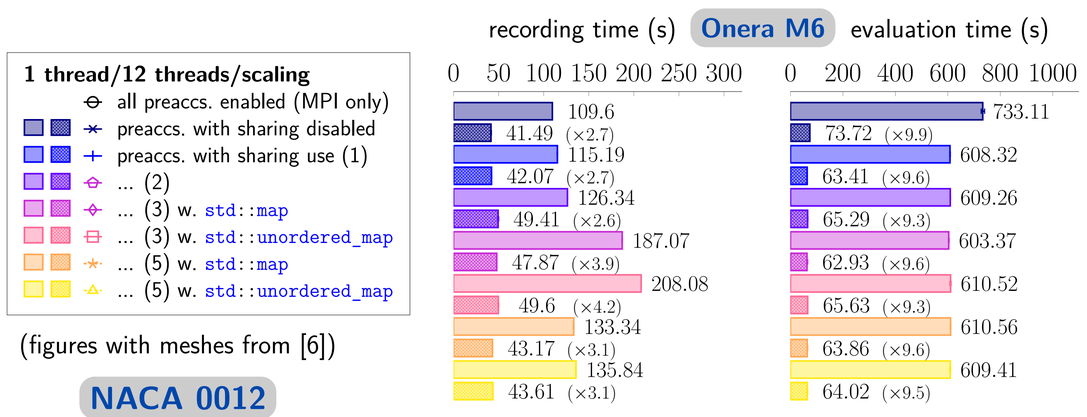


	runtime for single adjoint access	runtime for memory allocation (one thread)	memory consumption (all threads)	remarks/further costs
(1) large vector	$\mathcal{O}(1)$	$\mathcal{O}(i_\mathsf{max})$ (worst case)	$\mathcal{O}(i_\mathsf{max} \cdot T)$	vectors may persist across preaccs., reallocation not always required
(2) large vector with offset	$\mathcal{O}(1)$	$\mathcal{O}(i_\mathsf{max})$ (worst case)	$\mathcal{O}(i_\mathsf{max} \cdot T)$ (worst case)	same as (1), $\mathcal{O}(\lvert V_t\rvert)$ runtime to compute min, max
(3) map, `std::map`	$\mathcal{O}(\log \lvert V_t\rvert)$ (upper bound)	included in access times	$\mathcal{O}(\lvert V_t\rvert \cdot T)$	maps can be populated dynamically
(3) map, `std::unordered_map`	$\mathcal{O}(1)$ (average) $\mathcal{O}(\lvert V_t\rvert)$ (worst case)	included in access times	$\mathcal{O}(\lvert V_t\rvert \cdot T)$	same as (3) w. `std::map`, also applies to other maps below additional indirection, memory for both map and vector, compute $\lvert V_t\rvert$ (might be known)
(4) remap identifiers, `std::map`, small vector	$\mathcal{O}(\log \lvert V_t\rvert)$ (upper bound)	included in access times (map) $\mathcal{O}(\lvert V_t\rvert)$ (vector)	$\mathcal{O}(\lvert V_t\rvert \cdot T)$	
(4) remap identifiers, `std::unordered_map`, small vector	$\mathcal{O}(1)$ (average) $\mathcal{O}(\lvert V_t\rvert)$ (worst case)	included in access times (map) $\mathcal{O}(\lvert V_t\rvert)$ (vector)	$\mathcal{O}(\lvert V_t\rvert \cdot T)$	same as (4) w. `std::map`
(5) map to edit graph, `std::map`, small vector	$\mathcal{O}(1)$	$\mathcal{O}(\lvert V_t\rvert)$ (vector)	$\mathcal{O}(\lvert V_t\rvert \cdot T)$	editing has same map-related time complexity and temporary memory consumption as one evaluation in (4) w. `std::map`
(5) map to edit graph, `std::unordered_map`, small vector	$\mathcal{O}(1)$	$\mathcal{O}(\lvert V_t\rvert)$ (vector)	$\mathcal{O}(\lvert V_t\rvert \cdot T)$	analogous to (5) w. `std::map`, but w.r.t. (4) w. `std::unordered_map`

- (1) consumes too much memory, (2) could save memory but is often as bad in practice
- (3) has unfavourable runtime, (4) has no advantages compared to (3)
- (5) advances (4), combines advantages of (1) and (3) in case of multiple evaluations

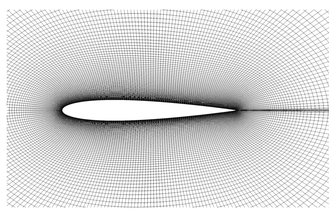
Benchmarking in SU2

- open-source multiphysics simulation-suite **SU2** [9, 10]
  - discrete adjoints by means of reverse AD [1, 2]
  - operator overloading (OO) AD with **CoDiPack** [12]
  - **MeDiPack** for OO AD of MPI
- **OpDiLib** provides OO AD of OpenMP [7, 3]
- our prior work enables OpenMP-parallel discrete adjoints in SU2 with OpDiLib [6, 5]
- **goal:** preaccs. with sharing disabled in [6], re-enable them using local adjoints [5, 4]
- benchmarks with three test cases from [6], Elwetritsch cluster at RPTU
  - aerodynamic lift as objective, mesh coordinates as parameters
  - SU2 uses **reverse accumulation**, an iterative AD approach for differentiating fixed-points [8]
  - primary recording, multiple evaluations for rev. acc.; secondary recording, 1 evaluation
  - Intel Xeon Gold 6126 (each 12 cores, 1 NUMA domain), dual-socket nodes
  - ⊖ is MPI-only, one proc. per core; others: 1 proc. per socket, 1 thread per core
  - (5) applied only to preaccs. with multiple evaluations, otherwise use corresponding variant of (3)



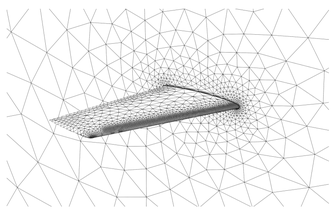
(figures with meshes from [6])

NACA 0012



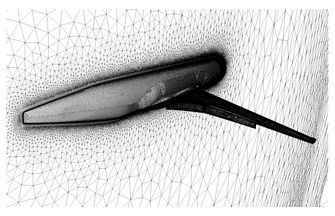
75140 quad.
300 fixed-point iter.

Onera M6

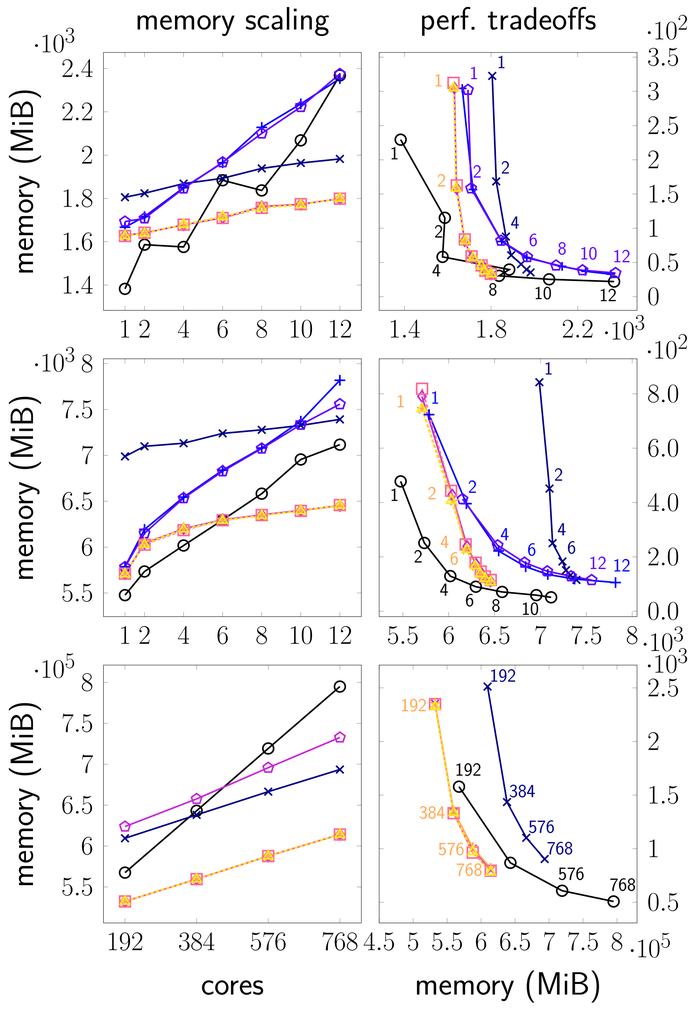


258969 tetr.
300 Krylov iter.

NASA HL-CRM



18 Mio. cells/8.3 Mio. nodes
1000 Krylov iter.



References

[1] Tim Albring, Max Sagebaum, and Nicolas R. Gauger. "Development of a Consistent Discrete Adjoint Solver in an Evolving Aerodynamic Design Framework". In: *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2015, pp. 1–14. DOI: 10.2514/6.2015-3240.

[2] Tim Albring, Max Sagebaum, and Nicolas R. Gauger. "Efficient Aerodynamic Design using the Discrete Adjoint Method in SU2". In: *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. 2016, pp. 1–15. DOI: 10.2514/6.2016-3518.

[3] Johannes Blühdorn and Nicolas R. Gauger. *Automatic Differentiation of OpenMP with OpDiLib*. NHR Conference '23 – Book of Abstracts (extended version), pp. 176–177. Poster contribution. Sept. 18, 2023. URL: https://www.nhr-verein.de/sites/default/files/2024-01/NHR%20Conference%202023%20-%20Book%20of%20Abstracts.pdf (visited on Aug. 13, 2024).

[4] Johannes Blühdorn and Nicolas R. Gauger. *Local Adjoints for Simultaneous Preaccumulations with Shared Inputs*. 2024. arXiv: 2405.07819v2 [cs.MS]. **Primary reference**.

[5] Johannes Blühdorn and Nicolas R. Gauger. *OpenMP-Parallel Discrete Adjoints in SU2 with OpDiLib*. NHR Conference '24. Poster contribution. Sept. 9, 2024.

[6] Johannes Blühdorn, Pedro Gomes, Max Aehle, and Nicolas R. Gauger. "Hybrid Parallel Discrete Adjoints in SU2". In: *Computers & Fluids* 289 (2025). Article No.: 106528, pp. 1–18. DOI: 10.1016/j.compfluid.2024.106528.

[7] Johannes Blühdorn, Max Sagebaum, and Nicolas R. Gauger. "Event-Based Automatic Differentiation of OpenMP with OpDiLib". In: *ACM Transactions on Mathematical Software* 49.3 (1 2023), pp. 1–31. DOI: 10.1145/3570159.

[8] Bruce Christianson. "Reverse accumulation and attractive fixed points". In: *Optimization Methods and Software* 3.4 (1994), pp. 311–326. DOI: 10.1080/10556789408805572.

[9] Thomas D. Economon, Francisco Palacios, Sean R. Copeland, Trent W. Lukaczyk, and Juan J. Alonso. "SU2: An Open-Source Suite for Multiphysics Simulation and Design". In: *AIAA Journal* 54.3 (2016), pp. 828–846. DOI: 10.2514/1.J053813.

[10] Pedro Gomes, Thomas D. Economon, and Rafael Palacios. "Sustainable High-Performance Optimizations in SU2". In: *AIAA Scitech 2021 Forum*. 2021, pp. 1–18. DOI: 10.2514/6.2021-0855.

[11] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Vol. 105. Philadelphia, PA, USA: SIAM, 2008. DOI: 10.1137/1.9780898717761.

[12] Max Sagebaum, Tim Albring, and Nicolas R. Gauger. "High-performance derivative computations using CoDiPack". In: *ACM Trans. Math. Softw.* 45.4 (Dec. 2019). 34. DOI: 10.1145/3356900.

Primary reference.
See also references cited therein.

Software

SU2 logo courtesy:
SU2 foundation,
https://su2foundation.org/wp-content/uploads/2019/10/SU2_Logo_eps.zip
(visited on Aug. 15, 2024)