OpenMP-Parallel Discrete Adjoints in SU2 with OpDiLib

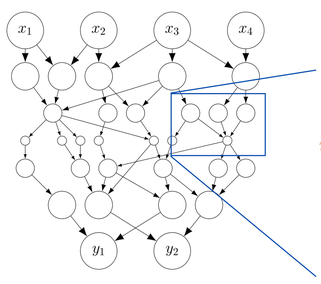
Johannes Blühdorn, Nicolas R. Gauger

Chair for Scientific Computing, University of Kaiserslautern-Landau (RPTU)

Context and Goals

- open-source multiphysics simulation-suite **SU2** [7]
  - SU2 features **discrete adjoints** by means of operator-overloading (OO) **automatic differentiation** (AD) [1, 2]
  - both primal and discrete adjoint solvers support **MPI** parallelism
  - primal solvers support **OpenMP** parallelism [8]
- **goal**: enable and improve OpenMP-parallel discrete adjoint solvers in SU2 [6, 4]
- apply **OpDiLib**, an add-on for OO-AD tools providing reverse AD of OpenMP [5, 3]

Automatic Differentiation



	Primal	Forward AD	Reverse AD
individual statement	$w = \phi(u_1, u_2, u_3)$	$\dot{w} = \Sigma_i \frac{\partial \phi}{\partial u_i} \dot{u}_i$	$\forall i\colon \bar{u}_i \mathrel{+}= \frac{\partial \phi}{\partial u_i} \bar{w}$
whole program	$y = F(x)$	$\dot{y} = \frac{\mathrm{d}}{\mathrm{d}x} F(x) \dot{x}$	$\bar{x} = \frac{\mathrm{d}}{\mathrm{d}x} F(x)^{\mathrm{T}} \bar{y}$

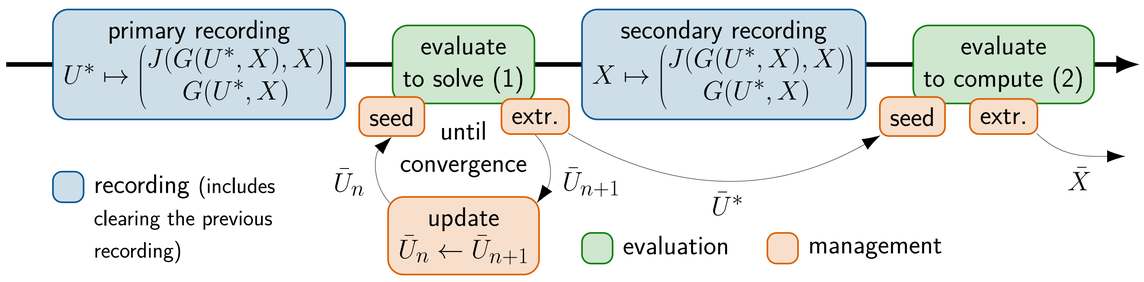
- computational graph with values as nodes and direct dependencies as edges
  - edges annotated with partials, nodes annotated with identifiers (virtual addresses)
  - identifiers address into memory for tangent values or adjoint values
- see [9] for AD in general, the above corresponds to Jacobian taping in CoDiPack [10]

Discrete Adjoints in SU2

- primal fixed-point equation $U = G(U, X)$ with parameters X, state U, and iterator G
- with primal solution U^* and objective function J, compute $\bar{X} = \frac{\mathrm{d}}{\mathrm{d}X} J(G(U^*, X), X)^{\mathrm{T}}$ by means of discrete adjoint equations

$$\bar{U} = \frac{\partial}{\partial U} J(G(U^*, X), X)^{\mathrm{T}} + \frac{\partial}{\partial U} G(U^*, X)^{\mathrm{T}} \bar{U}, \tag{1}$$

$$\bar{X} = \frac{\partial}{\partial X} J(G(U^*, X), X)^{\mathrm{T}} + \frac{\partial}{\partial X} G(U^*, X)^{\mathrm{T}} \bar{U} \tag{2}$$



Enabling OpenMP-Parallel Discrete Adjoints

Coupling with OpDiLib

- enables recording of OpenMP-parallel code
- provides corresponding parallel evaluation
- `SU2_OMP_*`/`END_SU2_OMP_*` macros
  - conform with OpDiLib's macro backend
- pair recording positions with OpDiLib states
- parallel external function treatment of LSEs

```
SU2_OMP_PARALLEL
{
   SU2_OMP_FOR_(SU2_NOWAIT)
   for (int i = 0; i < n; ++i)
   { ... }
   END_SU2_OMP_FOR
   ...
   SU2_OMP_BARRIER
   ...
}
END_SU2_OMP_PARALLEL
```

Exchange Identifier Management

- previous linear management strategy (LMS) assumes sequential recording
  - copies share identifiers, like references for AD
- transition to a parallel reuse management strategy
  - rewrite code that depends on properties of LMS

Parallelized AD Workflow

- revised storage layout of input/output identifiers
- parallel management
- parallel (re)computation of dependent quantities

Performance Improvements

Adjoint Vector Management

- CoDiPack's vector of adjoint variables is a shared resource
- mutual exclusion of resizing and reading/writing by shared mutex
  - less exclusive locking: explicit resizing, access without bounds checking
  - less shared locking: single lock for multiple subsequent lock-free accesses

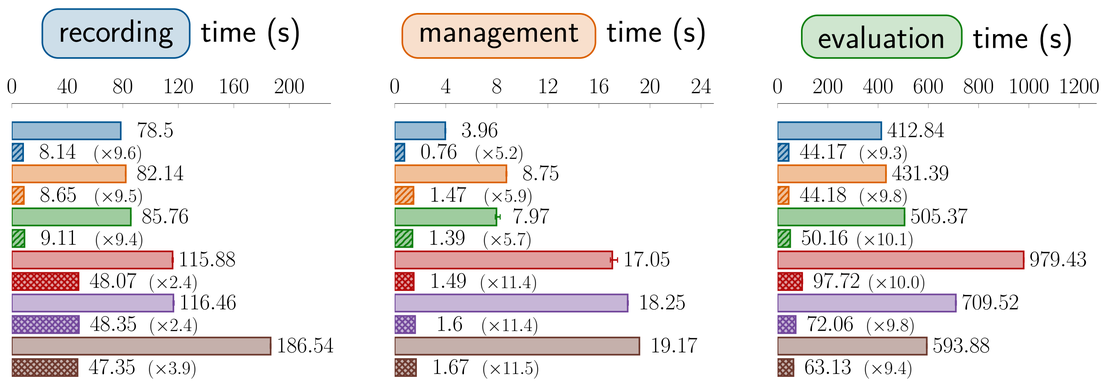
Shared Reading Optimization

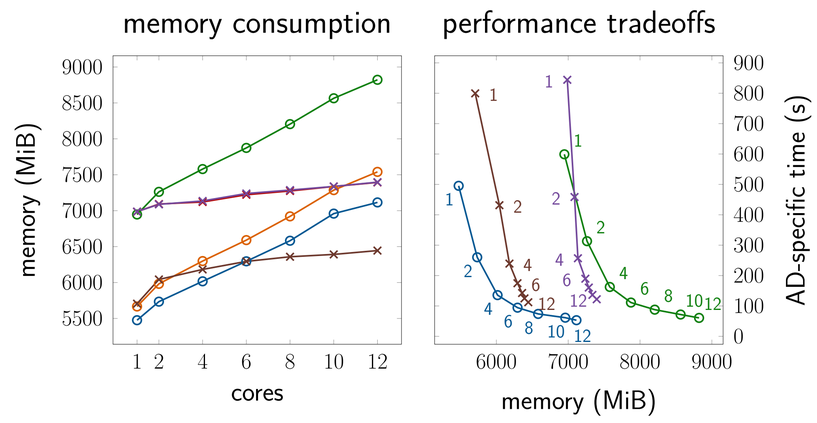
- non-exclusive primal reads become non-exclusive adjoint writes
- data races resolved by atomic updates on adjoint variables
- leverage developer knowledge: annotate code parts with exclusive read access
  - evaluations of those parts do not use atomics

Preaccumulations with Shared Inputs

- preaccumulations: early contractions of computational subgraphs [9]
- simultaneous thread-local preaccumulations with shared inputs result in data races [6, 4]
- default: disable these preaccumulations
- transition to thread-local adjoint variables for re-enabling these preaccumulations [4]
  - here: thread-local adjoints via `std::map<int, double>`

Onera M6 Benchmark





- primal solver: air flow around an Onera M6 wing
  - AoA 3.06°, Mach 0.8395, Re $11.72 \cdot 10^6$, L 0.64607 m
- RANS-SA, JST, Green-Gauss, FGMRES, ILU
- aerodynamic lift as objective function
- mesh coordinates as parameters
- solve (1) by 300 GMRES iterations
  - restart Krylov solver every 50^{th} iteration
- Skylake node of the Elwetritsch cluster at RPTU
  - Intel Xeon Gold 6126 (12 cores, 1 NUMA domain)

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: 2405.07819v1 [cs.MS]. **Primary reference**.

[5] 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.

[6] Johannes Blühdorn et al. *Hybrid Parallel Discrete Adjoints in SU2*. 2024. arXiv: 2405.06056v2 [cs.MS]. **Primary reference**.

[7] Thomas D. Economon et al. "SU2: An Open-Source Suite for Multiphysics Simulation and Design". In: *AIAA Journal* 54.3 (2016), pp. 828–846. DOI: 10.2514/1.J053813.

[8] 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.

[9] 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.

[10] 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 references.
See also references cited therein.

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