EuroLLVM 2025 - Berlin

MLIR Tensor Compiler

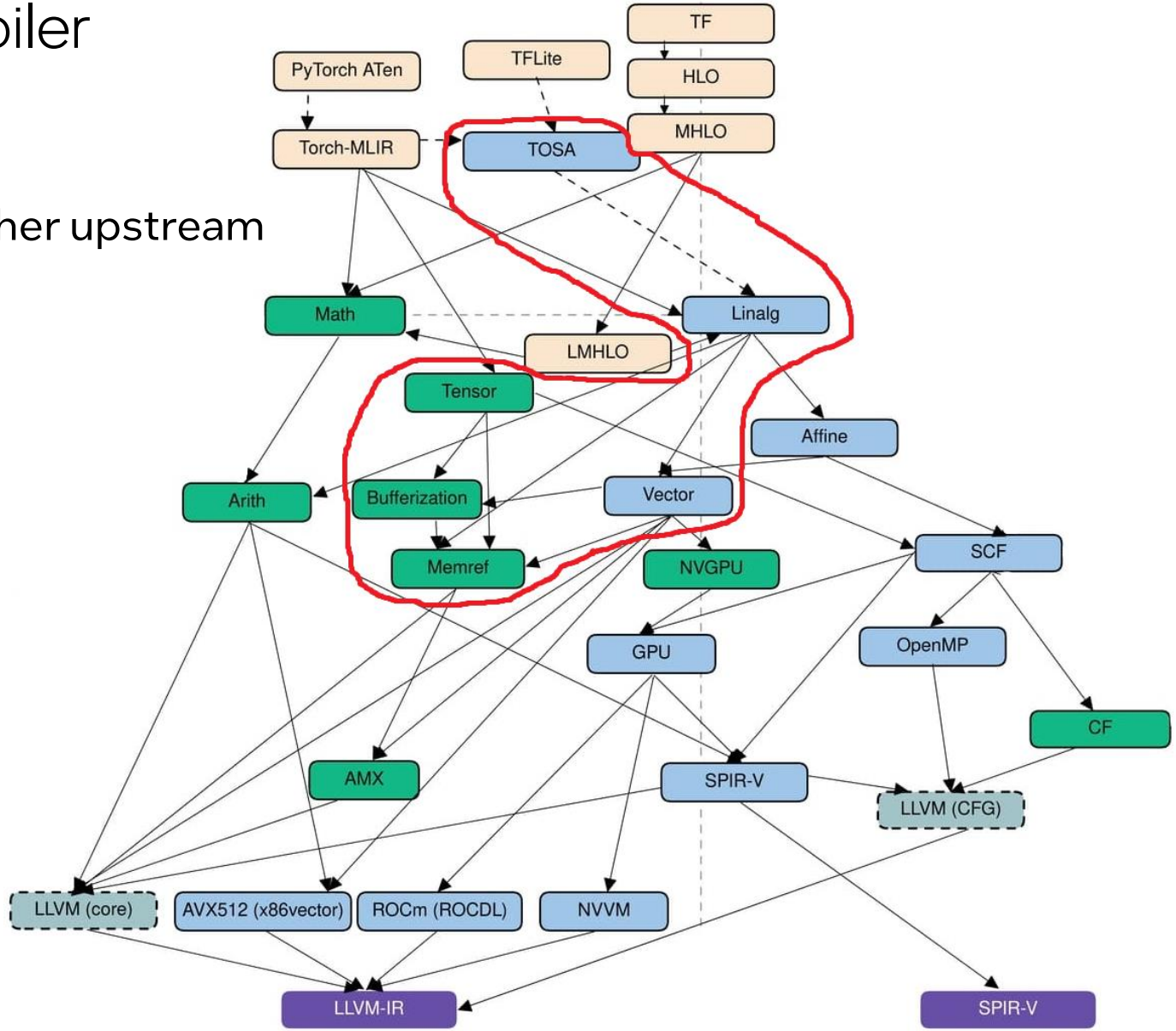
design group & charter update

Rolf Morel & Renato Golin

Upstream MLIR Tensor Compiler

- MLIR-based toolkit for ML compilers
  - Great value in developing this together upstream

- Main dialects:
  - linalg
  - tensor
  - memref
  - vector
  - bufferization
  - tosa



Original image credit: Quinn Dawkins & Alex Zinenko

MLIR organization – dialect groupings

- Originates with MLIR Governance efforts from last year
  - [RFC] MLIR Project Charter and Restructuring by Renato Golin, Stella Laurenzo, Chris Lattner, Alex Zinenko, Jacques Pienaar, Mehdi Amini
  - [Survey] MLIR Project Charter and Restructuring Survey by Renato
  - MLIR Organization & Charter by Renato, Stella, Alex, Jacques, Chris, and Andrzej Warzynski , Nicolas Vasilache, Mahesh Ravishankar

MLIR organization – dialect groupings

- Originates with MLIR Governance efforts from last year
  - [RFC] MLIR Project Charter and Restructuring by Renato Golin, Stella Laurenzo, Chris Lattner, Alex Zinenko, Jacques Pienaar, Mehdi Amini
  - [Survey] MLIR Project Charter and Restructuring Survey by Renato
  - MLIR Organization & Charter by Renato, Stella, Alex, Jacques, Chris, and Andrzej Warzynski , Nicolas Vasilache, Mahesh Ravishankar

- Tensor Compiler dialect grouping
  - **linalg**, **tensor**, **memref**, **vector**, **bufferization**, **tosa**
  - Own community of stakeholders
  - In need of an overarching charter
  - Clear governance: upstream consensus

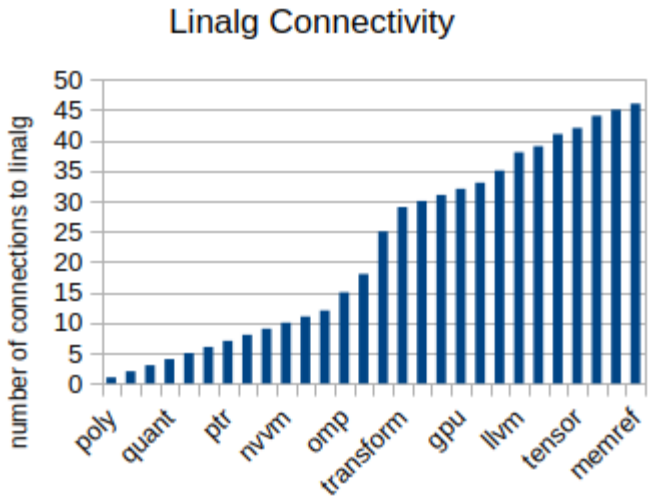
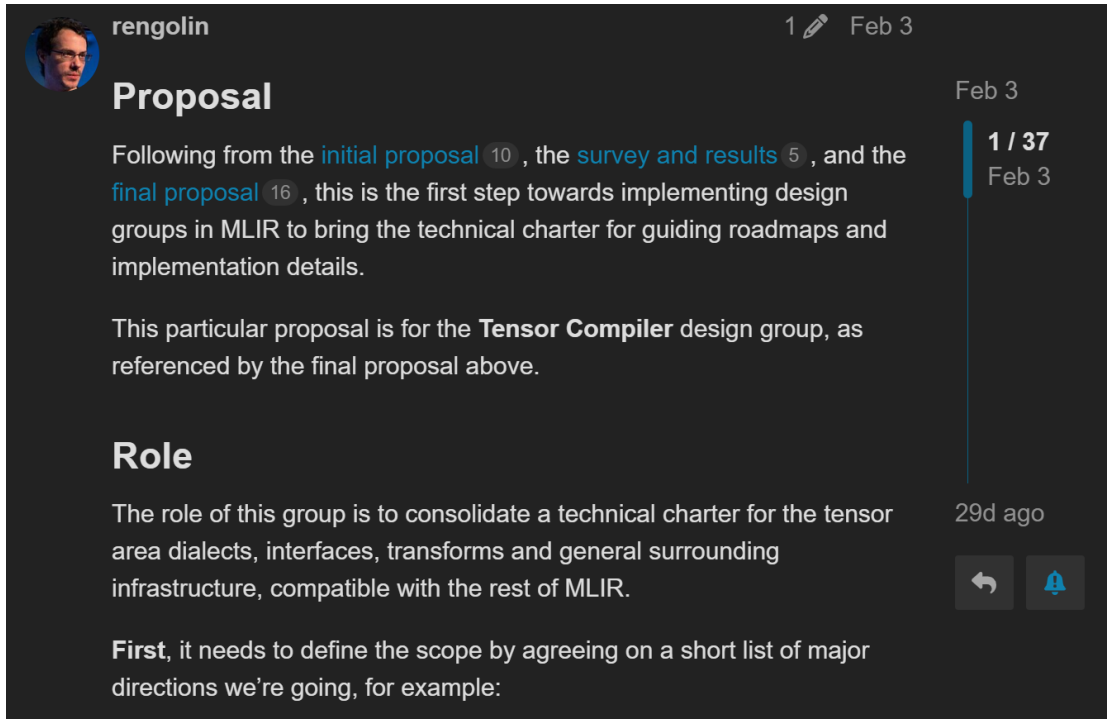


Image credit: Renato Golin

Tensor Compiler Design Group

- MLIR Tensor Compiler Design Group by Renato

  Goal: charter across relevant dialects, interfaces, transforms, surrounding infrastructure



Tensor Compiler Design Group

- MLIR Tensor Compiler Design Group by Renato

  Goal: charter across relevant dialects, interfaces, transforms, surrounding infrastructure

  Scope

  – linear algebra – tensor semantics – bufferization – memref semantics – vector semantics

  – canonicalization (op aliasing within linalg) – ingress & egress – dialect design – flows

Tensor Compiler Design Group

- MLIR Tensor Compiler Design Group by Renato

  Goal: charter across relevant dialects, interfaces, transforms, surrounding infrastructure

  Scope

  – linear algebra – tensor semantics – bufferization – memref semantics – vector semantics

  – canonicalization (op aliasing within linalg) – ingress & egress – dialect design – flows

  Documentation updates

  – Consolidate conflicting roadmaps

  – Update outdated rationale docs

  – Start an overarching charter

Tensor Compiler Design Group

- MLIR Tensor Compiler Design Group by Renato

  Goal: charter across relevant dialects, interfaces, transforms, surrounding infrastructure

  Scope

  – linear algebra – tensor semantics – bufferization – memref semantics – vector semantics

  – canonicalization (op aliasing within linalg) – ingress & egress – dialect design – flows

  Documentation updates

  – Consolidate conflicting roadmaps

  – Update outdated rationale docs

  – Start an overarching charter

  Infrastructure for distributed usage

  – Devise a path for more flexibility for users (also across upstream projects)

  – E.g., easier dialect extensions, canonicalized transform requirements, better coverage

Tensor Compiler Design Group – Members

Volunteers from active contributors and representative stakeholders

Alex Zinenko Brium	Renato Golin Intel	Jacques Pienaar Google	Matthias Springer Nvidia
Quinn Dawkins AMD	Javed Absar Qualcomm	Jakub Kuderski AMD	Suraj Sudhir Arm
Rolf Morel Intel	Andrzej Warzynski Arm	Diego Caballero Nvidia	Kunwar Grover AMD

Tensor Compiler Design Group – Members

Volunteers from active contributors and representative stakeholders

Alex Zinenko Brium	***Renato Golin*** Intel	***Jacques Pienaar*** Google	***Matthias Springer*** Nvidia
Quinn Dawkins AMD	Javed Absar Qualcomm	Jakub Kuderski AMD	Suraj Sudhir Arm
Rolf Morel Intel	Andrzej Warzynski Arm	Diego Caballero Nvidia	Kunwar Grover AMD

In bold: MLIR Area Team, a distinct governance effort

Tensor Compiler Design Group – Members

Volunteers fr

Alex Zinenko
Brium

Quinn Dawkins
AMD

Rolf Morel
Intel

atthias Springer
Nvidia

Suraj Sudhir
Arm

Kunwar Grover
AMD

Modular

Democratizing AI Compute, Part 8:
What about the MLIR compiler infrastructure?

CHRIS LATTNER

A New Hope: Improved MLIR Governance

The tensions have simmered for years—and they're deeply felt across the broader LLVM and MLIR communities.

Fortunately, **there's a new hope**: LLVM is a meritocratic community with a long track record of aligning engineers—even when their companies are at war in the market. The MLIR community is filled with amazing engineers who have poured years of their hearts and souls into improving the project to work through these challenges, and progress is now happening!

MLIR now has a new **Area Team** to help guide its evolution, along with a **new organizational structure and charter** and **governance group**. The charter defines separate area groups: MLIR Core (the domain-independent infrastructure), and the dialects (like the machine learning-specific pieces). I am extremely thankful to everyone who is spending time to improve MLIR and work through these issues—such work has a profound impact on everyone building into the ecosystem as well as the downstream users.

**In bold*: MLIR Area Team, a distinct governance effort

Tensor Compiler Design Group – Modus Operandi

- Regular meetings
  - Bi-weekly group meetings
    - Agenda & notes: MLIR Tensor Compiler Design Group - Meetings
    - … only had three (3) so far
  - First **O**pen **D**esign **M**eeting scheduled late April (date TBC)

Tensor Compiler Design Group – Modus Operandi

- Regular meetings
  - Bi-weekly group meetings
    - Agenda & notes: MLIR Tensor Compiler Design Group - Meetings
    - … only had three (3) so far
  - First **O**pen **D**esign **M**eeting scheduled late April (date TBC)
- In practice, members bring in-the-pipeline, community-relevant topics
  - In essence *pre-RFCs* — workgroup provides space for quick iteration
    - Rapid top-of-mind responses for low overhead feedback
  - Next: posted as RFC, with same consensus process as any other RFC

Tensor Compiler Design Group – Modus Operandi

- Regular meetings
  - Bi-weekly group meetings
    - Agenda & notes: MLIR Tensor Compiler Design Group - Meetings
    - … only had three (3) so far
  - First **O**pen **D**esign **M**eeting scheduled late April (date TBC)
- In practice, members bring in-the-pipeline, community-relevant topics
  - In essence *pre-RFCs* — workgroup provides space for quick iteration
    - Rapid top-of-mind responses for low overhead feedback
  - Next: posted as RFC, with same consensus process as any other RFC
- All workgroup-generated documentation is public

MLIR Tensor Compiler Design Group - Overview document

Tensor Compiler Design Group – Progress on Vector

MLIR Tensor Compiler Design Group - Vector Dialect: Refactoring + Re-design ideas

RFCs which benefitted from live discussion:

- [RFC] Allow pointers as element type of `vector`
  - Upshot: VectorElementTypeInterface with semantics of only allowing “atomic” elements
- [RFC] Improving gather codegen for Vector Dialect
  - Addresses abstraction gap which lead to early loss of structured indexing
- [RFC] Generalize tiling to operate on ShapedType
  - Proposes extending infrastructure for tiling/blocking beyond linalg-on-tensor/memref
- In-the-pipeline RFC on reducing references to LLVM LangRef in vector dialect docs

Tensor Compiler – other recent significant changes

- Transpose attribute for Linalg matmul operations
  - Linalg.matmul (and batch_matmul) now have an affine_maps attribute
- Introduce linalg.contract : $D[\mathbf{J}] = \left(\bigoplus_{((I^A \cup I^B) \setminus \mathbf{J})} A[I^A] * B[I^B] \right) \oplus C[\mathbf{J}]$
  - Op generalizing all contraction ops (e.g. matmul variants and matvec and dot and …)
- Extend Linalg elemwise named ops semantics
  - linalg.elementwise with affine_maps replacing linalg.elem_wise_{unary,binary}
- Move `tensor.pack` and `tensor.unpack` into Linalg
  - Fix layering issue; facilitate interactions with linalg ops; allow for packing on memrefs

Topics we are looking to make progress on

1. [RFC] Should we restrict the usage of 0-D vectors in the Vector dialect?
2. Remove ops with overlapping functionality from Vector, e.g. extractelement -> extract
3. Various issues in Vector regarding consistent naming and semantics
4. Pros & cons of splitting vector dialect into high-level and low-level parts
5. Vector vs Tensor types – another go at clarifying their distinctive roles

Topics we are looking to make progress on

1. [RFC] Should we restrict the usage of 0-D vectors in the Vector dialect?
2. Remove ops with overlapping functionality from Vector, e.g. extractelement -> extract
3. Various issues in Vector regarding consistent naming and semantics
4. Pros & cons of splitting vector dialect into high-level and low-level parts
5. Vector vs Tensor types – another go at clarifying their distinctive roles
6. Revisit how to attach layouts to tensors & vectors
7. Enshrine in the charter the significance of projected permutation affine_maps
8. Pick up stalled progress on common-ing up linalg conv ops
9. Compute type on linalg ops given correspondence with imperfect loop nest
10. Op aliasing in linalg, e.g. linalg.matmul vs linalg.contract vs linalg.generic
    - Equiv. class perspective on matching w.r.t. [RFC] Linalg operation tree

Tensor Compiler Design Group – next steps

- **More focus on documentation updates**
  - Many dialect-level documents are outdated, as are the rationale ones
  - We would love help with this!

Tensor Compiler Design Group – next steps

- **More focus on documentation updates**
  - Many dialect-level documents are outdated, as are the rationale ones
  - We would love help with this!
- **Start formulating the Tensor Compiler-spanning, cross-dialect charter**
  - New documents laying out intended coherent usage across dialects

Tensor Compiler Design Group – next steps

- **More focus on documentation updates**
  - Many dialect-level documents are outdated, as are the rationale ones
  - We would love help with this!
- **Start formulating the Tensor Compiler-spanning, cross-dialect charter**
  - New documents laying out intended coherent usage across dialects
- **Start codifying flows through the Tensor Compiler**
  - E.g., integration tests spanning just the Tensor Compiler dialects
  - In addition to explicitly described flows in the charter

Tensor Compiler Design Group – **get in touch!**

Alex Zinenko
(@ftynse)

Renato Golin
(@rengolin)

Jacques Pienaar
(@jpienaar)

Matthias Springer
(@matthias-springer)

Quinn Dawkins
(@qed)

Javed Absar
(@javedabsar)

Jakub Kuderski
(@kuhar)

Suraj Sudhir
(@sjarus)

Rolf Morel
(@rolfmorel)

Andrzej Warzynski
(@banach-space)

Diego Caballero
(@dcaballe)

Kunwar Grover
(@groverkss)