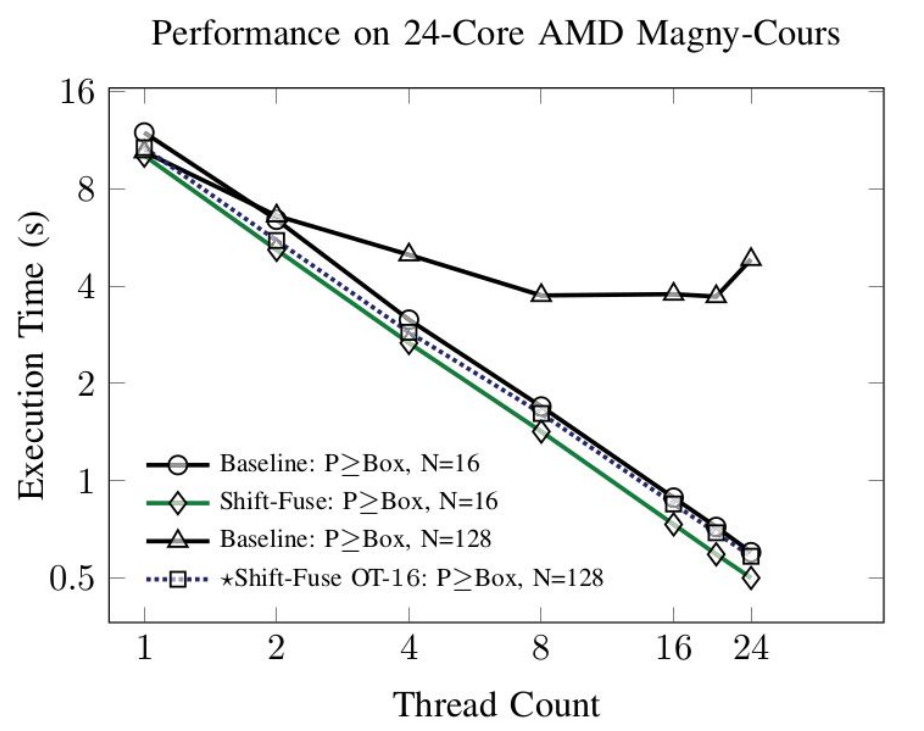
Identifying and Scheduling Loop Chains Using Directives

Ian J. Bertolacci - University of Arizona
Michelle Mills Strout - University of Arizona
Stephen Guzik - Colorado State University
Jordan Riley - Colorado State University
Catherine Olschanowsky - Boise State University

Problem Space

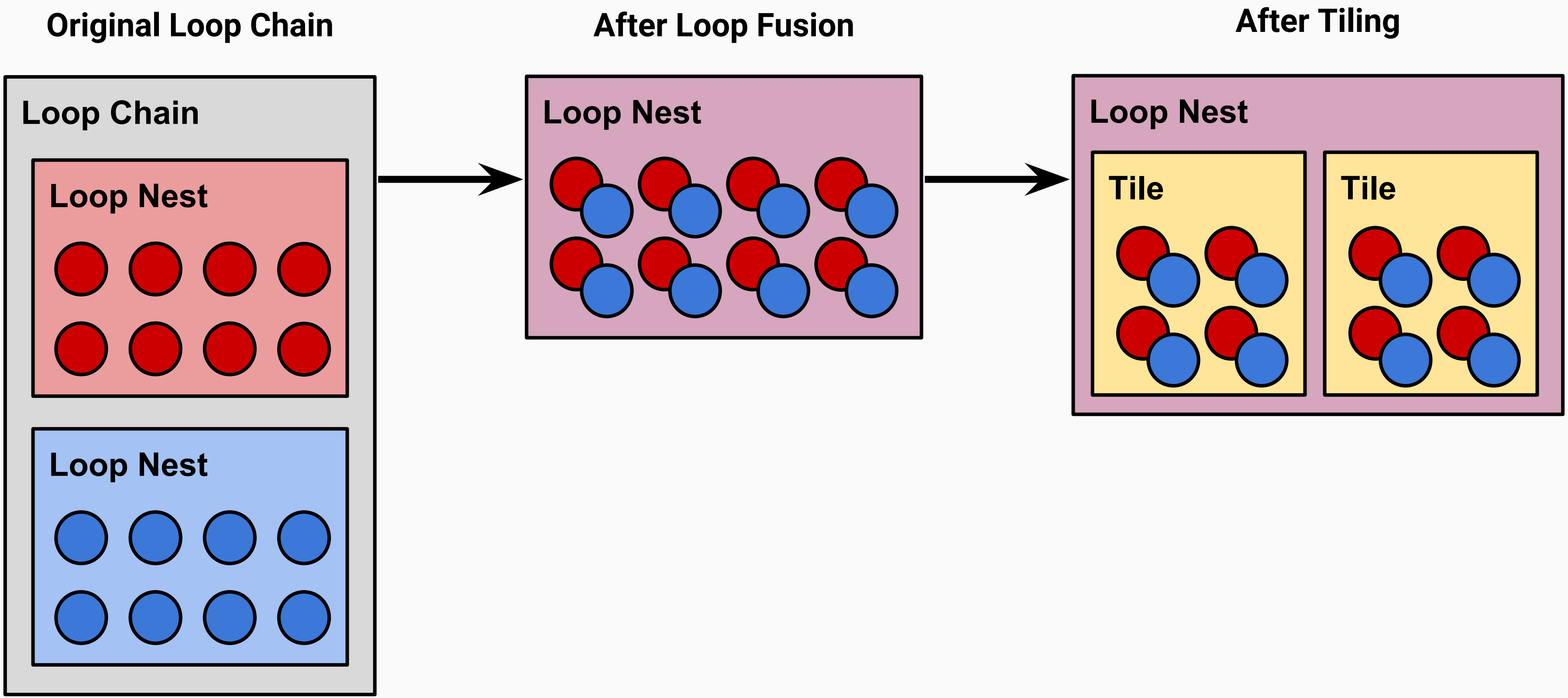
- Stencil computations are a large part of physical simulation applications.
- Plenty of opportunities for parallelism.
- Memory pressure causes poor performance.
- Scheduling holistically, *across* adjacent loop nests, provides balance between parallelism and data locality [1].

[1] C. Olschanowsky, M. M. Strout, S. Guzik, J. Loffeld, and J. Hittinger, “A Study on Balancing Parallelism, Data Locality, and Recomputation in Existing PDE Solvers,” in *To be published in The IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014.



Performance results of loop chain optimizations in mini-flux-div benchmark [1]. Baseline is the original highly optimized implementation.

Loop Chain Optimizations



Specific Problem

1. Currently, Loop chain optimizations laboriously performed by hand.
2. Fully automating loop chain optimizations is not realistic.
3. Rewriting in a domain specific language may not be feasible.

Can we find a middle ground?

Yes!

Using annotations on existing code to inform compiler.

Our Contributions

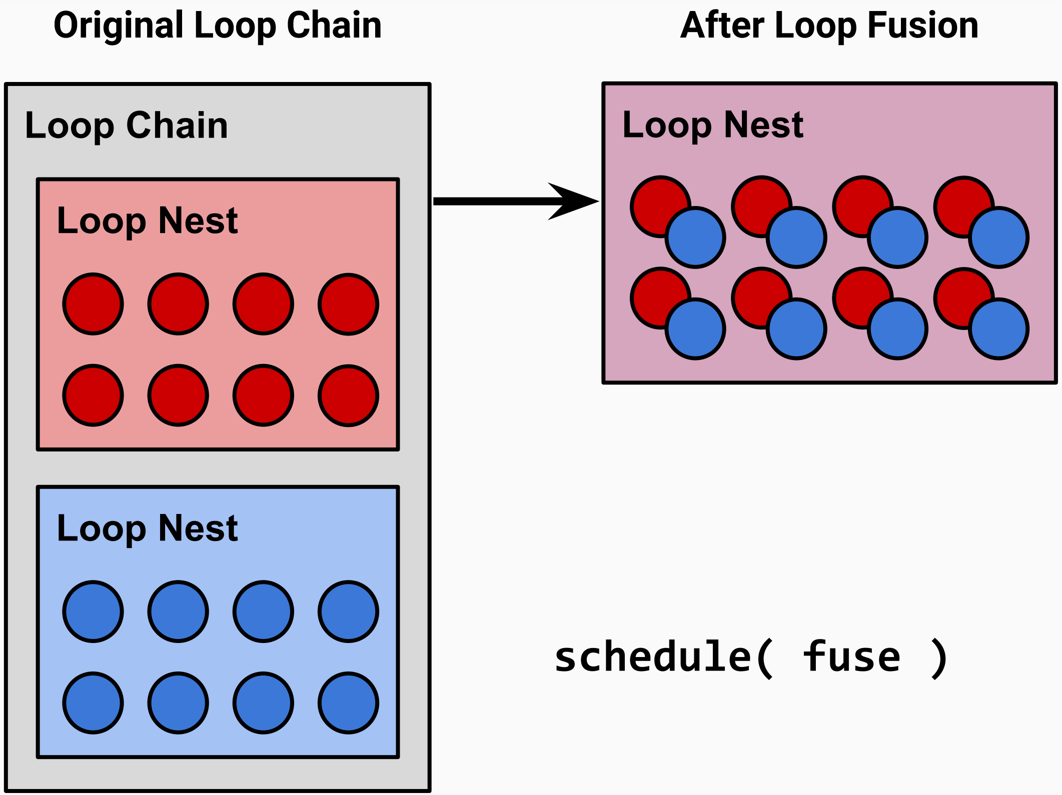
1. Developed annotations to describe loop chains and specify loop chain optimizations.

2. Early implementation of a loop chain optimization compiler pass utilizing these annotations.

Annotations

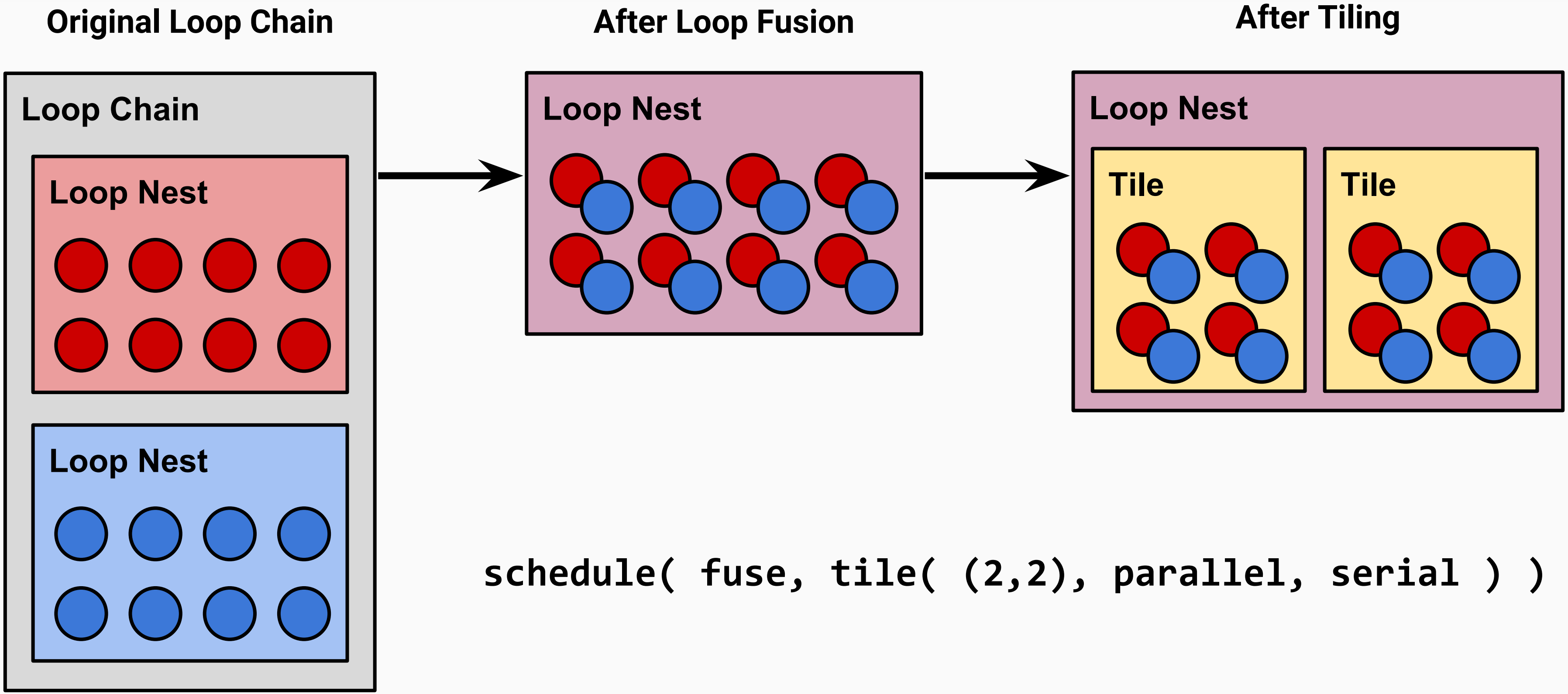
- **`#pragma ...`**
  - **`... loopchain schedule(`** *`[transformations]`* **`)`**
    - Denote a block as a loop chain.
    - Specifies the ways in which loops should be transformed.
      - Currently have fusion and tiling transformations.
      - **`schedule( fuse, tile( (10,20), parallel, serial) )`**
  - **`... for domain(`** *`[domain of nest]`* **`) with`** *`[access patterns]`*
    - Denote a loop nest in the parent loop chain.
    - Specifies the bounds of the nest.
    - Specifies how the loop nest reads and writes data at each iteration.

Schedule Example



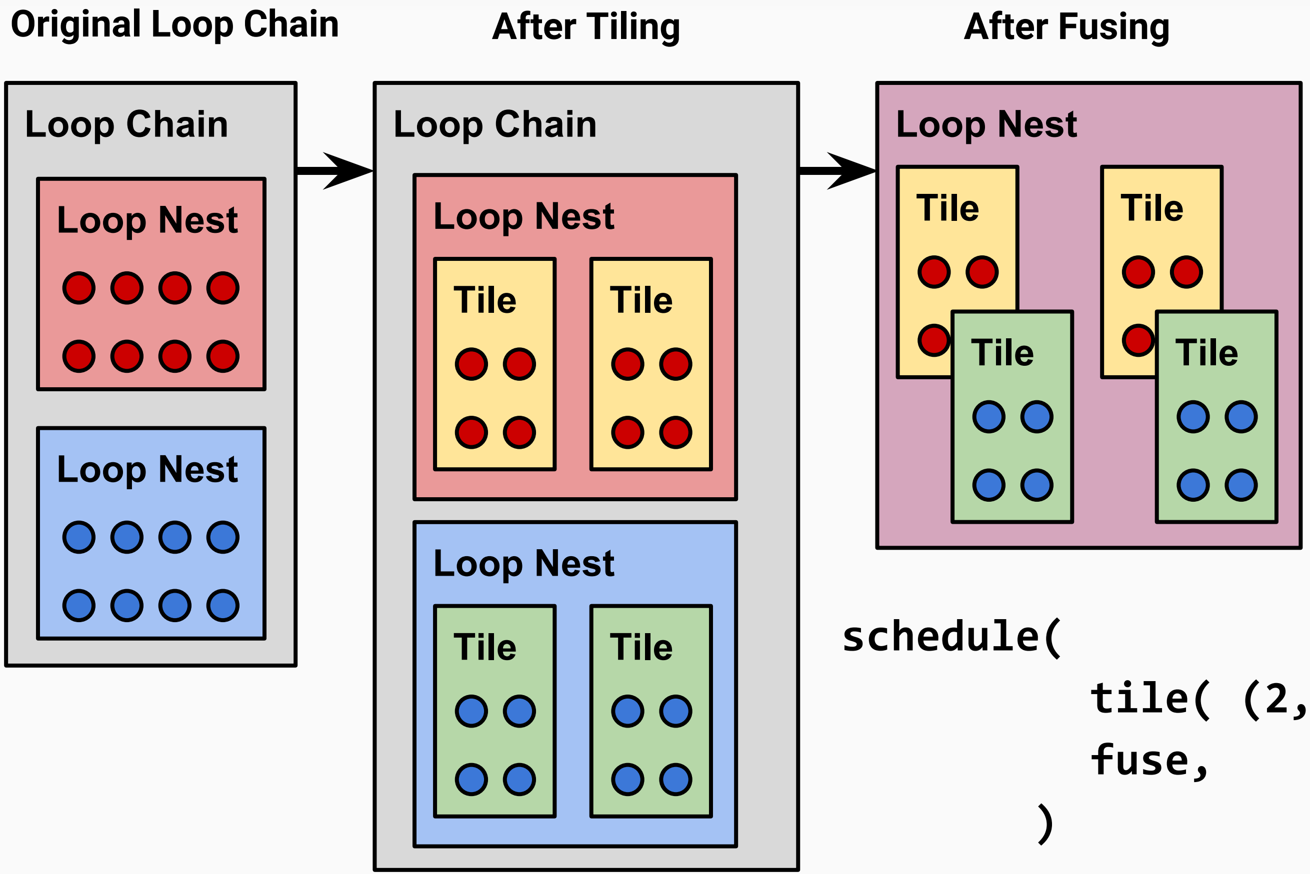

```
schedule( fuse )
```

Schedule Example




```
schedule( fuse, tile( (2,2), parallel, serial ) )
```

Schedule Example

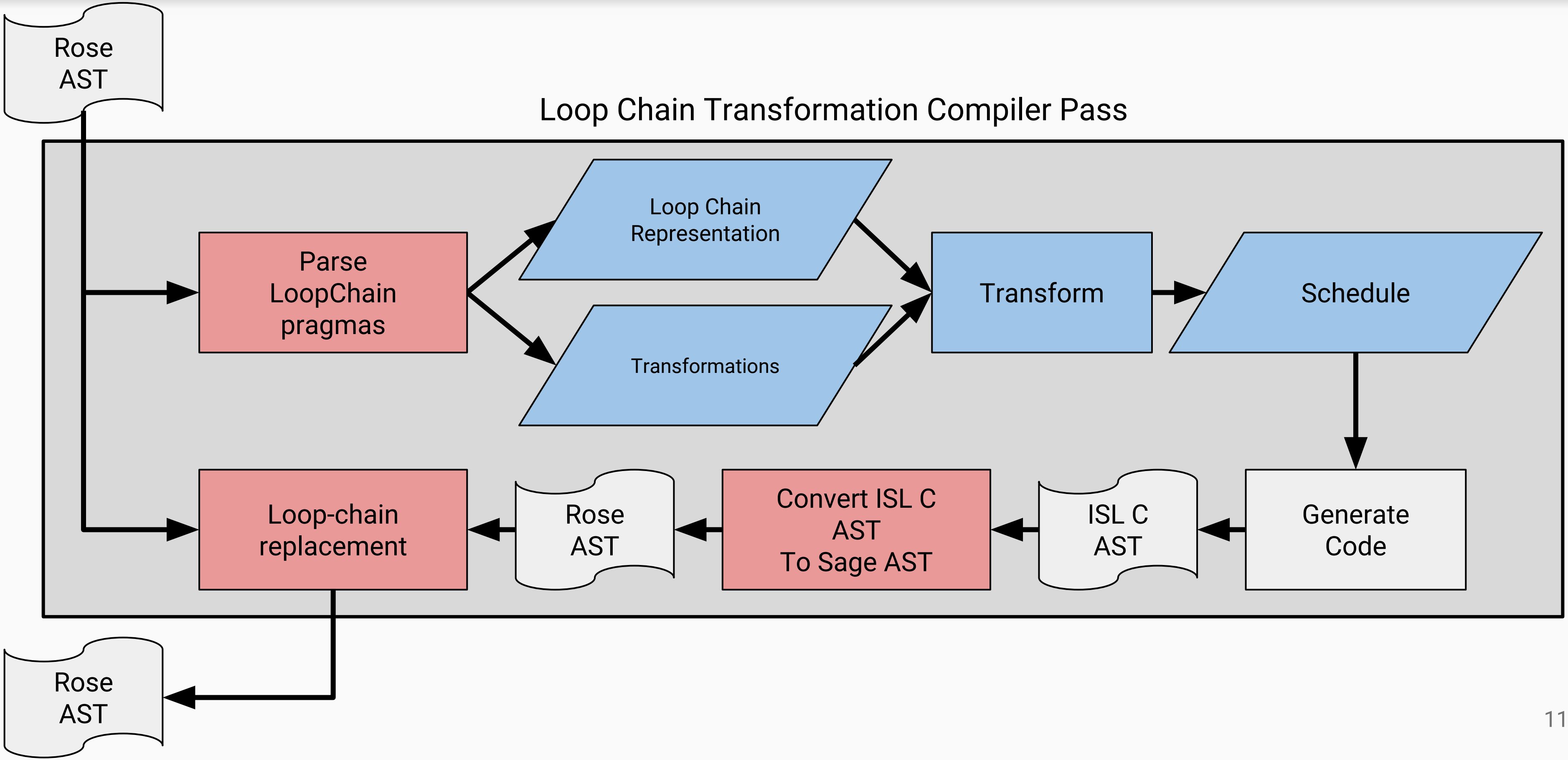



```
schedule(
        tile( (2,2), serial, serial ),
        fuse,
      )
```

Compiler Pass Toolbox

- Rose Compiler Framework
  - Performs all parsing and compilation.
  - API for performing code synthesis and manipulation.
- LoopChainIR
  - Our intermediate representation for loop chains and transformations.
- Integer Set Library (ISL)
  - Takes loop domains and transformation functions specified in LoopChainIR performs the mathematical transformations and code generation.

Compiler Pass Overview



Conclusion

- We are working towards using code annotations to enable automated optimizations on loop chains.
- In this work we have provided:
  - Code annotations to describe loop chains, their access into data, and the ways they are to be scheduled.
  - An early implementation of a compiler pass that uses these annotations to perform code transformations.
- Future work
  - Automate shifting before loop fusions.
  - Additional schedule operations, overlapped tiling in particular.