Driving HPC Parallel Optimizations with DSL

Improve resource allocations

Maël MARTIN, Patrick CARRIBAULT, Hugo TABOADA

Summary

1. Context
2. Characteristics & Implementation
3. Resource allocation
4. Conclusion

Context

- Exaflopic supercomputers have diversified architectures
- *Domain Specific Language/Abstraction* (DSL/DSA) allows to abstract parallelism and hardware specificity. The need to exploit such specificity is shifted to the compilers and runtimes
  - NabLab is a DSL used at CEA for stencil codes
- Compilers are based on *Intermediate Representations* (IR), their internal representation of the code. They can have specific properties:
  - SSA, parallelism information, or a functional code structure representation
- Related work:
  - MLIR[1] and xDSL[4] are infrastructures that aim to lower the cost of implementing new DSL/DSA
  - Firedrake[2] and Devito[3] are DSL/DSA used in numeric simulations

Characteristics & Implementation

- Restricted expressivity of the code
- Three types of regularities :
  - Statically regular, Dynamically regular, Dynamically irregular

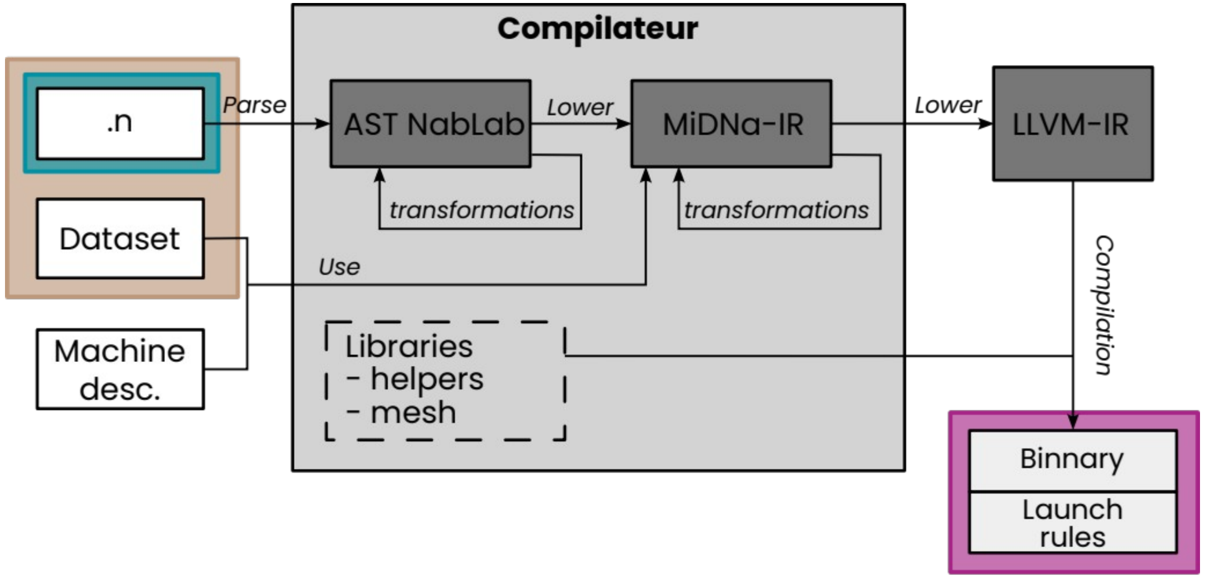


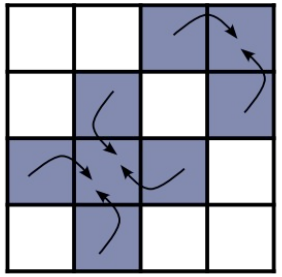
Fig 1. NabLab compiler

Resource allocation

- Find the optimal usage of allocated resources
  - Memory footprint
  - cores count prediction
  - target execution time

Resource allocation

- Find the optimal usage of allocated resources
  - Memory footprint
  - cores count prediction
  - target execution time



```
for i in cells(), {
  A{i} = Sum{j in neighbor(i)}(
    compute(B{j}, C{i})
  );
}
```

Parallel part

Incompressible part

$$S_C(s) = \frac{1}{1-C+\frac{C}{s}}$$

$$C_n \,/\, S_{C_n}(s) \geq n \times S_\infty(s), n \in]0,1[$$

Fig 2.a. Execution model of the compute loops

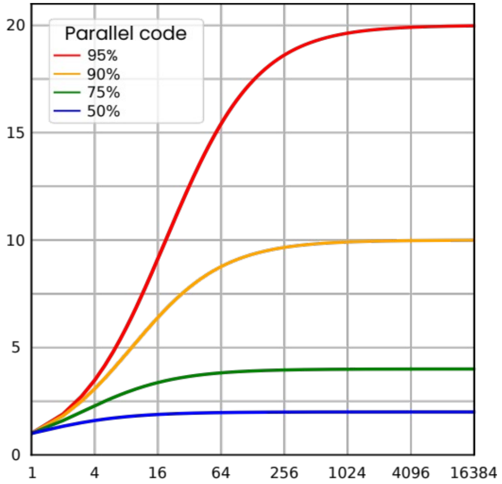


Fig 2.b. Amdahl law for the scaling of each loop; gives the speedup depending on the core count

Resource allocation

- Find the optimal usage of allocated resources
  - Memory footprint
  - cores count prediction
  - target execution time

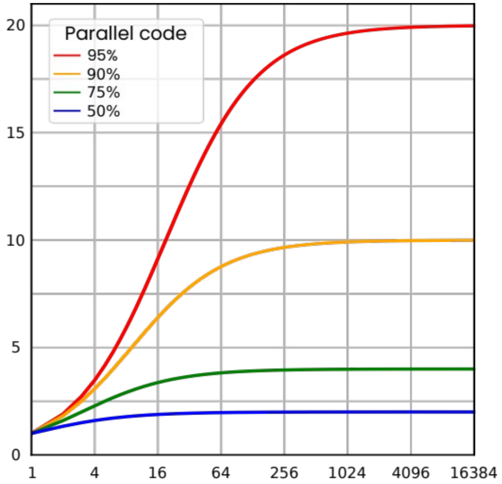


Fig 2.b. Amdahl law for the scaling of each loop; gives the speedup depending on the core count

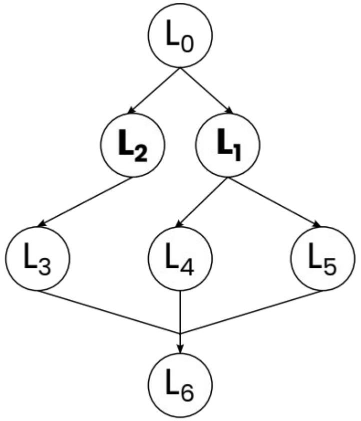


Fig 2.c. Scheduling of the loops depending on their scaling in the DAG of the loops

Resource allocation

- Find the optimal usage of allocated resources
  - Memory footprint
  - cores count prediction
  - target execution time

- Schedule of L_1 and L_2:
  - The loops are 75% parallel
  - Speedup: 3.9 on 128cores, 3.8 on 64cores
  - Exec. times: 2.5s on 128 cores, 2.6 on 64cores
  - Total exec. times: 5s or 2.6s

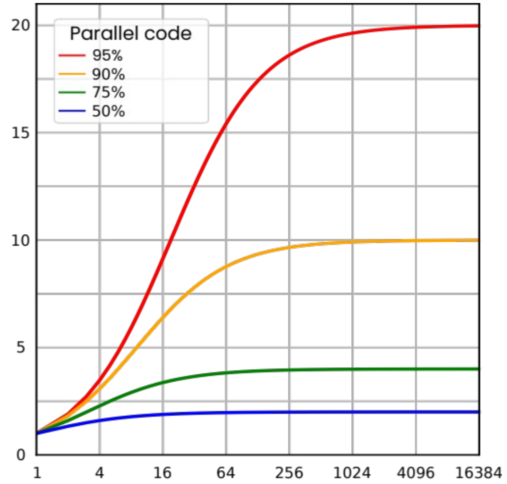


Fig 2.b. Amdahl law for the scaling of each loop; gives the speedup depending on the core count

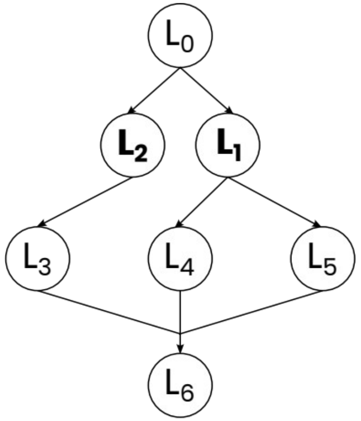


Fig 2.c. Scheduling of the loops depending on their scaling in the DAG of the loops

Conclusion

- Exploration of static scheduling of loops to minimize global execution time
- Implementation of a compiler from the grammar to parallel code
  - Keep track of information from the parsing of the language to the code generation
  - Pass hints to the runtime for a better execution
- Resource allocation deduced from the algorithmic complexity of the application
- Future works
  - Add distributed and heterogeneous code generation
  - Add helper threads in the code generation and resource allocation algorithm
  - Better guide the LLVM optimizer with vectorization hints
  - Better guide the used runtimes

- **[1]** Lattner, Chris, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis, Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, et Oleksandr Zinenko. « MLIR: Scaling Compiler Infrastructure for Domain Specific Computation ». In 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2-14. Seoul, Korea (South): IEEE, 2021. .
- **[2]** Gheorghe-Teodor Bercea, Andrew T. T. McRae, David A. Ham, Lawrence Mitchell, Florian Rathgeber, Luigi Nardi, Fabio Luporini, and Paul H. J. Kelly. « A structure-exploiting numbering algorithm for finite elements on extruded meshes, and its performance evaluation in firedrake ». In Geoscientific Model Development, 9(10):3803–3815, 2016.
- **[3]** Luporini, Fabio and Louboutin, Mathias and Lange, Michael and Kukreja, Navjot and Witte, Philipp and Hückelheim, Jan and Yount, Charles and Kelly, Paul H. J. and Herrmann, Felix J. and Gorman, Gerard J. « Architecture and Performance of Devito, a System for Automated Stencil Computation ». In 2020 ACM Trans. Math. Softw.
- **[4]** Nick Brown, Tobias Grosser, Mathieu Fehr, Michel Steuwer, Paul Kelly. « xDSL: A common compiler ecosystem for domainspecific languages ». In SC22 Poster session. https://sc22.supercomputing.org/proceedings/tech_poster/poster_files/rpost133s3-file3.pdf,

Maël MARTIN

Mael.MARTIN@cea.fr