Machine Specific Symbolic Code Generation for Computational Science Applications

Robert King(Student)* Mentors: Hari Sundar †, Milinda Fernando,‡
*u1001542@utah.edu, †hari@cs.utah.edu, ‡milinda@cs.utah.edu,

I. Introduction

New discoveries in science and engineering are primarily driven by numerical simulations of underlying governing equations specially when the physical experiments become infeasible. High performance computing (HPC) is widely used to perform these simulations efficiently. When moving towards exascale computing, HPC clusters are moving towards increased heterogeneity and frequent change in architectures. Ability to utilize modern and future HPC clusters effectively is highly depend on performance portability and adaptation to new architectures. Manually written codes to evaluate the main computational kernels lack portability, prone to human errors, ability to perform code optimizations due to the complexity of the underlying equations. In this work we present a symbolic code generation framework, which generates architecture optimized code for different platforms. As the driving application we primarily use computational relativity where computations of Einstein equations become complicated due to the presence of curvature in spacetime. But the algorithms presented in this work, is applicable to generate code for any underlying applications.

The key contributions of this work include:

- **Symbolic inferace** : The presented framework is based on `SymPy` with additional modules written to handle complicated partial differential equations (PDEs).
- **Equations** $\rightarrow$ **Graphs** : The symbolically written equations are converted to a computational graph, which enables to perform architecture (cache, register optimizations) and language specific (SIMD vectorization, CUDA) optimizations.
- **Common Subexpression Elimination (CSE)**: By computing common subexpressions, we can reduce the number of compute operations needed, by storing them in temory variables.

II. Background & Related Work

In this section, we present a brief introduction on the driving application, where the symbolic code generation is deployed. The recent discovery of gravitational waves (GWs) in 2015, has excited the computational relativity community. Numerically computed GWs are important to perform verification and matched filtering for the massive amount of data generated by GW detectors. Computational relativity is primaraly focused on evolving 3+1 (space + time) decomposition of 4 Einstein equations for a specified initial condition. In this work we use commonly used , formulation of Einstein equations.The Einstein equations are a set of non-linear, coupled, partial differential equations. On discretization, one can end up with 24 or more equations with thousands of terms. Writing, optimizing and maintaining code for this is very challenging. Sustainability and keeping it relevant for new architectural changes are additional difficulties. To address these issues, use our symbolic code generation framework, to generate architecture optimized codes to compute the BSSN equations (see Figure **??**).

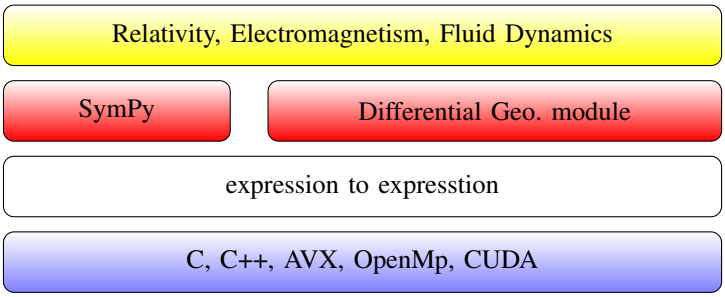

Fig. 1: put a caption

III. Methodology

The symbolic code generation framework is only the initial step towards connecting symbolic equations to architecture specific compute codes (see Figure **??**). By representing the equations as directed acyclic graphs (DAG) enable to exploit computation to match the programming language (AVX, CUDA, OpenMp) and architecture (CPU, GPU) specifications which result in faster and efficient codes. These specifications are met by performing *expression to expression* transformations such that transformations are mathematically equivalent.

A. Subtree isomorphism

B. Staging computations

IV. Results

V. Future Work

$$
\begin{aligned}
\partial_t \alpha &= \mathcal{L}_\beta \alpha - 2\alpha K, \\
\partial_t \beta^i &= \lambda_2 \beta^j \, \partial_j \beta^i + \frac{3}{4} f(\alpha) B^i \\
\partial_t B^i &= \partial_t \tilde{\Gamma}^i - \eta B^i + \lambda_3 \beta^j \, \partial_j B^i - \lambda_4 \beta^j \, \partial_j \tilde{\Gamma}^i \\
\partial_t \tilde{\gamma}_{ij} &= \mathcal{L}_\beta \tilde{\gamma}_{ij} - 2\alpha \tilde{A}_{ij}, \\
\partial_t \chi &= \mathcal{L}_\beta \chi + \frac{2}{3} \chi \left(\alpha K - \partial_a \beta^a \right) \\
\partial_t \tilde{A}_{ij} &= \mathcal{L}_\beta \tilde{A}_{ij} + \chi \left(-D_i D_j \alpha + \alpha R_{ij} \right)^{TF} + \\
&\quad \alpha \left(K \tilde{A}_{ij} - 2 \tilde{A}_{ik} \tilde{A}^k_j \right), \\
\partial_t K &= \beta^k \partial_k K - D^i D_i \alpha + \\
&\quad \alpha \left(\tilde{A}_{ij} \tilde{A}^{ij} + \frac{1}{3} K^2 \right), \\
\partial_t \tilde{\Gamma}^i &= \tilde{\gamma}^{jk} \partial_j \partial_k \beta^i + \frac{1}{3} \tilde{\gamma}^{ij} \partial_j \partial_k \beta^k + \beta^j \partial_j \tilde{\Gamma}^i - \\
&\quad \tilde{\Gamma}^j \partial_j \beta^i + \frac{2}{3} \tilde{\Gamma}^i \partial_j \beta^j - 2 \tilde{A}^{ij} \partial_j \alpha + \\
&\quad 2\alpha \left(\tilde{\Gamma}^i{}_{jk} \tilde{A}^{jk} + 6 \tilde{A}^{ij} \partial_j \phi - \frac{2}{3} \tilde{\gamma}^{ij} \partial_j K \right)
\end{aligned}
$$

```
from dendro_sym import *
a_rhs = dendro.Lie(b, a) - 2*a*K

b_rhs = [3/4 * f(a) * B[i] +
l2*vec_j_del_j(b, b[i]) for i in e_i]

B_rhs = [Gt_rhs[i] - eta * B[i] +
        l3 * vec_j_del_j(b, B[i]) -
        l4 * vec_j_del_j(b, Gt[i])
        for i in e_i]

gt_rhs =  dendro.Lie(b, gt) - 2*a*At

chi_rhs = dendro.Lie(b, chi) +
 2/3*chi*(a*K - del_j(b))

At_rhs = dendro.Lie(b, At) + chi *
 dendro.TF(-DiDj(a) + a*dendro.Ricci) +
 a*(K*At -2*At_ikAtKj)

K_rhs = vec_k_del_k(K) - DIDi(a) +
 a*(1/3*K*K + A_ij_A_IJ(At))
```

Fig. 2: The left panel shows the BSSNKO formulation of the Einstein equations. These are tensor equations, with indices $i, j, \ldots$ taking the values $1, 2, 3$. On the right we show the DENDRO_sym code for these equations. DENDRO_sym uses SymPy and other tools to generate optimized C++ code to evaluate the equations. Note that $\mathcal{L}_\beta$, D, ∂ denote Lie derivative, covariant derivative and partial derivative respectively, and we have excluded $\partial_t \Gamma^i$ from DENDRO_sym to save space. (See [**?**], [**?**] for more information about the equations and the differential operators.)