ProtoX: A First Look

Het Mankad*, Sanil Rao*, Phillip Colella†, Brian Van Straalen†, Franz Franchetti*

* Carnegie Mellon University, † Lawrence Berkeley National Laboratory

Problem

Stencil Operations: A key component in numerical solutions to partial differential equations (PDEs).

Proto: It is a domain specific library written in C++ that provides a high level of abstraction for solving PDEs using various numerical methods.

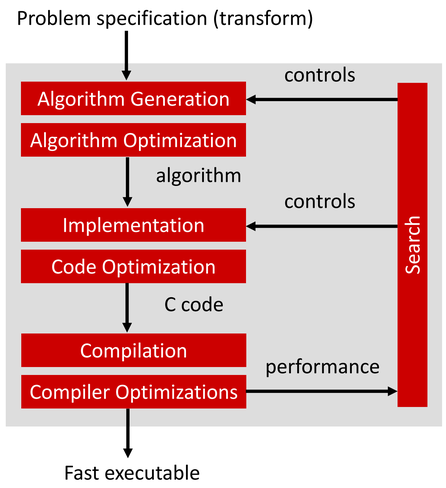
- **Shortcoming:** Abstraction fusion is something no compiler can easily perform.
- **Our Goal:** To interpret Proto as a Domain Specific Language with the help of SPIRAL [1],[2] and obtain better performance.

SPIRAL

Spiral:
Complete automation of the implementation and optimization task

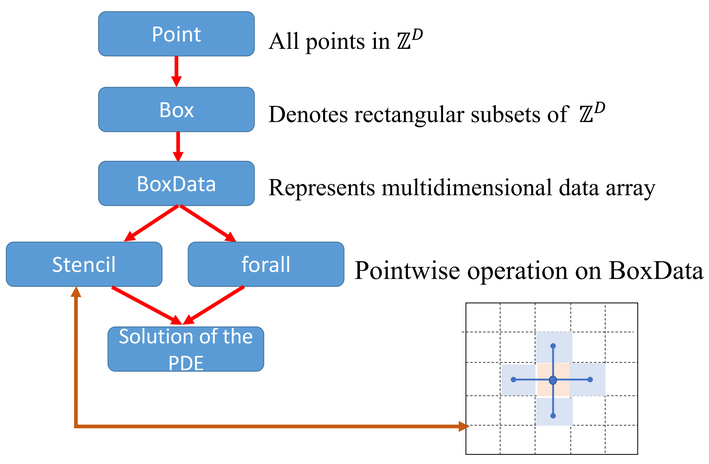
Basic idea:
Declarative representation of algorithms using the Operator Language (OL)

Rewriting systems to generate and optimize algorithms



Proto

Design layout and class structure in Proto



Acknowledgment

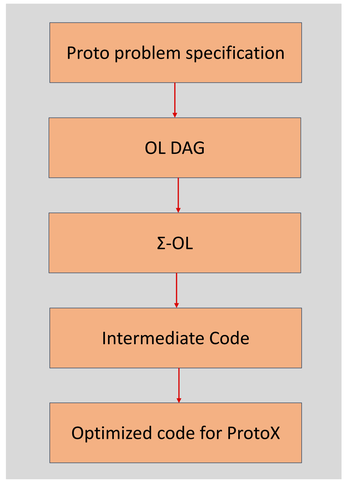
This project is funded by the DOE Office of Advanced Scientific Computing Research the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

ProtoX

Proto Frontend:
Proto source library interpreted as an embedded DSL

SPIRAL Backend:
Proto dataflow fed into SPIRAL as OL. It is then broken down into loop based Σ-OL that is translated into optimized C++ source code.

ProtoX:
Optimized code linked into Proto code to be executed in place of original specification.



ProtoX: Example

2D Euler equations for Gas Dynamics

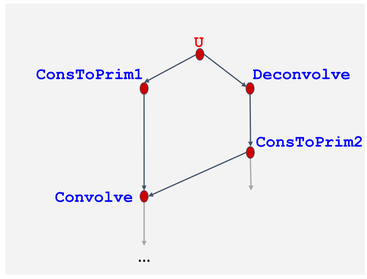
The 2D Euler equation used in gas dynamics takes the form of a conservation equation [3], $\frac{\partial U}{\partial t} + \nabla \cdot \vec{F}(U) = 0$, where

$$U_t = \begin{bmatrix} \rho \\ \rho u_x \\ \rho u_y \\ E \end{bmatrix}_t , \quad F(U)_x = \begin{bmatrix} \rho u_x \\ \rho u_x^2 + p \\ \rho u_x u_y \\ (E+p)u_x \end{bmatrix}_x , \quad F(U)_y = \begin{bmatrix} \rho u_y \\ \rho u_x u_y \\ \rho u_y^2 + p \\ (E+p)u_y \end{bmatrix}_y$$

and the total energy is $E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u_x^2 + u_y^2)$.

DAG for the spatial discretization of 2D Euler equations

```
inline void operator()(
   BoxData<T, NUMCOMPS>&a_Rhs,
   std::array<BoxData<T,NUMCOMPS>, DIM>&
   a_fluxes,const BoxData<T, NUMCOMPS>&
   a_U, T a_scale = 1.0) const
   {  …
     Vector W_bar = forallOp<double,
       NUMCOMPS>(ctoprmnum, "consToPrim",
       f_consToPrim, a_U, gamma);
     Vector U = Operator::deconvolve(a_U);
     Vector W = forallOp<double,
       NUMCOMPS>(ctoprmnum, "consToPrim",
       f_consToPrim, U, gamma);
     Vector W_ave = Operator::_convolve(W,
       W_bar);
       …
   }
```



OL DAG in SPIRAL

$$\text{BoxOPEuler}_{n,\gamma} \rightarrow \cdots (\text{Convolve}_n) \circ (\text{I}_{4n^2} \oplus \text{ConsToPrim2}_{n,\gamma}) \circ \begin{bmatrix} \text{ConsToPrim1}_{n,\gamma} \\ \text{Deconvolve}_n \end{bmatrix} \circ \text{X}_{i \times n \times n}^{i=1,\cdots,4}$$

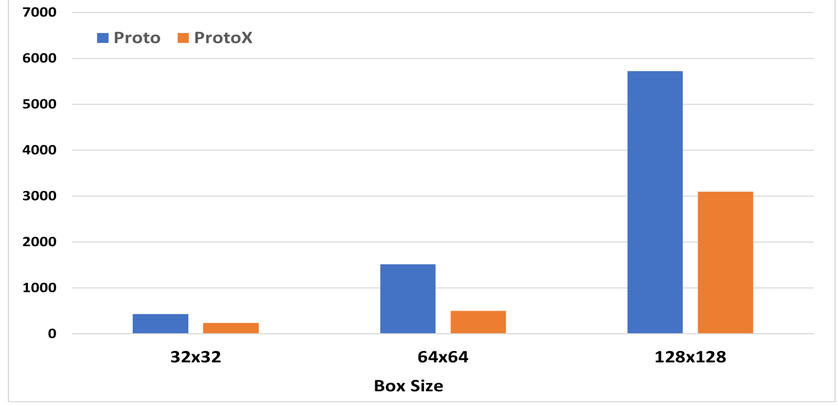
Σ − OL in SPIRAL

$$\left(\sum_{j=0}^{N-1} \text{S}_{r_j} \cdot \text{Convolve}_j \cdot \text{G}_{s_j}\right)\left(\left(\sum_{j=0}^{N-1} \text{S}_{p_j} \cdot \text{ConsToPrim1}_j \cdot \text{G}_{f \circ g}\right) \times \left(\sum_{j=0}^{N-1} \text{S}_{u_j} \text{ConsToPrim2}_j \cdot \text{Deconvolve}_j \text{G}_{g \circ f}\right)\right)$$

ProtoX Sample Fused Code

```
void full_eulerv3(double  *Y, const double  *X, double gamma1, double a_scale1, double dx1, double  *wavespeed, double  *retval1) {
  static double T1[12176];
  // ConsToPrim 1
  for(int i1 = 0; i1 <= 1599; i1++) {
    T1[i1] = X[i1];  T1[(i1 + 1600)]  = (X[(i1 + 1600)] / X[i1]);  T1[(i1 + 3200)]  = (X[(i1 + 3200)] / X[i1]); double a233 = (gamma1 - 1.0);
    double a234 = (0.5*a233);
    T1[(i1 + 4800)] = ((a233*X[(i1 + 4800)]) - (((0.5*a233)*((T1[(i1 + 1600)]*T1[(i1 + 1600)])*X[i1])) + ((0.5*a233)*((T1[(i1 + 3200)] *T1[(i1 +
     3200)])*X[i1])));
  }
  // Deconvolve +  ConsToPrim 2
  for(int i2 = 0; i2 <= 1443; i2++) {
    double a371, a372, s93, s94, s95, s96, s97, s98, t112;  nt b47 = ((40*(i2 / 38)) + (i2 % 38));
    s93 = X[(b47 + 41)]; s94 = X[(b47 + 1641)]; s95 = X[(b47 + 3241)]; s96 = X[(b47 + 4841)];
    t112 = (s93 - (0.041666666666666664*((X[(b47 + 1)] - (4.0*s93)) + X[(b47 + 40)] + X[(b47 + 42)] + X[(b47 + 81)])));
    T1[(i2 + 6400)] = t112;
    s97 = ((s94 - (0.041666666666666664*((X[(b47 + 1601)] - (4.0*s94)) + X[(b47 + 1640)] + X[(b47 + 1642)] + X[(b47 + 1681)]))) / t112);
    T1[(i2 + 7844)] = s97;
    s98 = ((s95 - (0.041666666666666664*((X[(b47 + 3201)] - (4.0*s95)) + X[(b47 + 3240)] + X[(b47 + 3242)] + X[(b47 + 3281)]))) / t112);
    T1[(i2 + 9288)] = s98;
    T1[(i2 + 10732)] = (((gamma1 - 1.0)*(s96 - (0.041666666666666664*((X[(b47 + 4801)] - (4.0*s96)) + X[(b47 + 4840)] + X[(b47 + 4842)] +
     X[(b47 + 4881)])))) - (((0.5*(gamma1 -  1.0))*((s97*s97)*t112)) + (a372*((s98*s98)*t112))));
  }
  // Convolve
  for(int i4 = 0; i4 <= 1443; i4++) {
    int b94 = ((40*(i4 / 38)) + (i4 % 38));
    T83[i4] = ((0.041666666666666664*((T1[(b94 + 1)] - (4.0*T1[(b94 + 41)])) + T1[(b94 + 40)] + T1[(b94 + 42)] + T1[(b94 + 81)])) + T1[(i4 +
     6400)]);
    T83[(i4 + 1444)] = ((0.041666666666666664*((T1[(b94 + 1601)] - (4.0*T1[(b94 + 1641)])) + T1[(b94 + 1640)] + T1[(b94 + 1642)] + T1[(b94
     + 1681)])) +  T1[(i4 + 7844)]);
    T83[(i4 + 2888)] = ((0.041666666666666664*((T1[(b94 + 3201)] - (4.0*T1[(b94 + 3241)])) + T1[(b94 + 3240)] + T1[(b94 + 3242)] + T1[(b94
     + 3281)])) +  T1[(i4 + 9288)]);
    T83[(i4 + 4332)] = ((0.041666666666666664*((T1[(b94 + 4801)] - (4.0*T1[(b94 + 4841)])) + T1[(b94 + 4840)] + T1[(b94 + 4842)] + T1[(b94
     + 4881)])) +  T1[(i4 + 10732)]);
    }
}
```

ProtoX optimized 2D Euler achieves up to **4x** performance improvement over the reference implementation



Conclusion & Future Work

- We demonstrate ProtoX, a system that interprets Proto as a Domain Specific Language using the 2D Euler equations as an example.
- Up to 4X improvement can be seen in the initial results. Efforts are being made for further optimization.
- The future goal is to add more targets and make ProtoX interoperable with FFTX [4] to do cross library optimization.

References

[1] M. Puschel et al., "SPIRAL: Code Generation for DSP Transforms," in Proceedings of the IEEE, vol. 93, no. 2, pp. 232-275, Feb. 2005.
[2] Franchetti et al., "Formal Loop merging for signal transforms", in Proceedings of ACM SIGPLAN, 40, 6 , pp. 315-326, June 2005.
[3]P. McCorquodale et. al., "A high order finite volume method for conservationlaws on locally refined grid" , Communications in Applied Mathematics and Computational Science, vol. 6, No.1, 2011
[4] F. Franchetti *et al.*, "FFTX and SpectralPack: A First Look," 2018 IEEE 25th International Conference on High Performance Computing Workshops, 2018, pp. 18-27