EVPFFTX: A First Look at FFTX Applications in Material Science

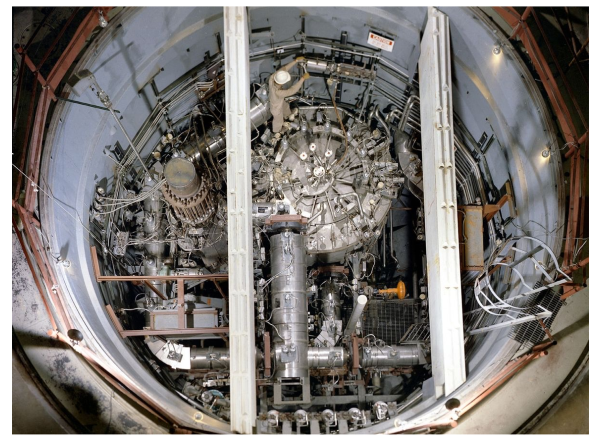
H. Mankad*, A. Rovinelli †, M. Zecevic †, P. McCorquodale‡, F. Franchetti*, N. Zhang*, S. Rao*, R. A. Lebensohn†, L. Capolungo†

* Carnegie Mellon University, † Los Alamos National Laboratory, ‡ Lawrence Berkeley National Laboratory

Problem

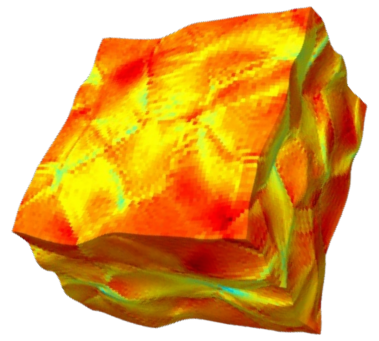
Molten Salt Reactor: Uses molten fluoride salts as fuel/primary coolant.

- Advantages: Cheaper, safer and can generate huge amount of energy and produce less waste.
- Challenge: Corrosivity and to study the effects of salt on different materials in a lab environment.



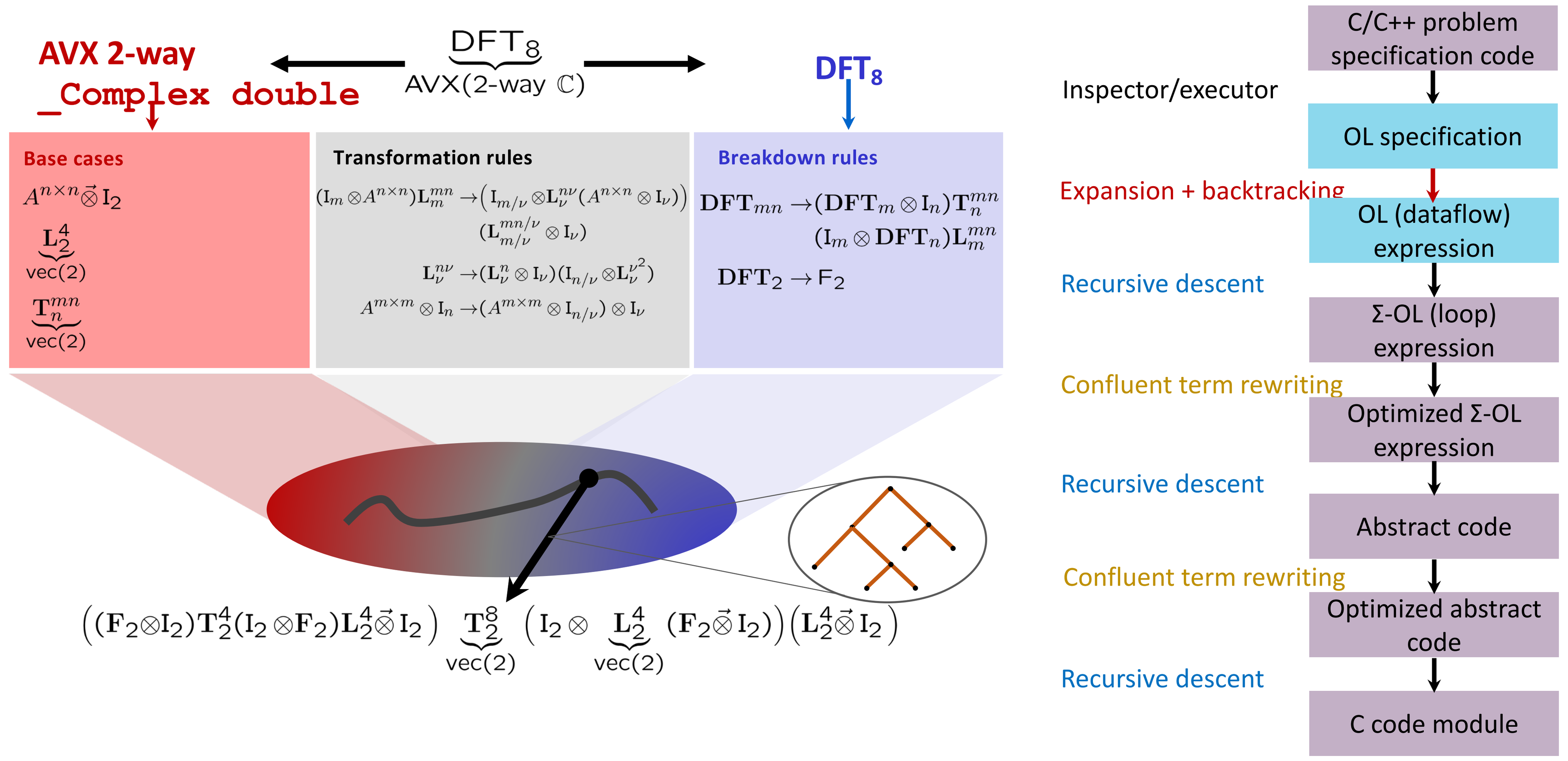
Elasto-Viscoplastic FFT (EVPFFT) [1] is an algorithm that helps study these effects on such microstructures.

- FORTRAN based code that studies polycrystals using FFTs.
- Goal: To boost the current performance of EVPFFT code.



EVPFFTX: An FFTX [3] based EVPFFT code that uses SPIRAL generated architecture specific optimized code to improve its current performance

SPIRAL



25 years of encoding domain knowledge:
Code generation/synthesis and autotuning as rule-based AI system

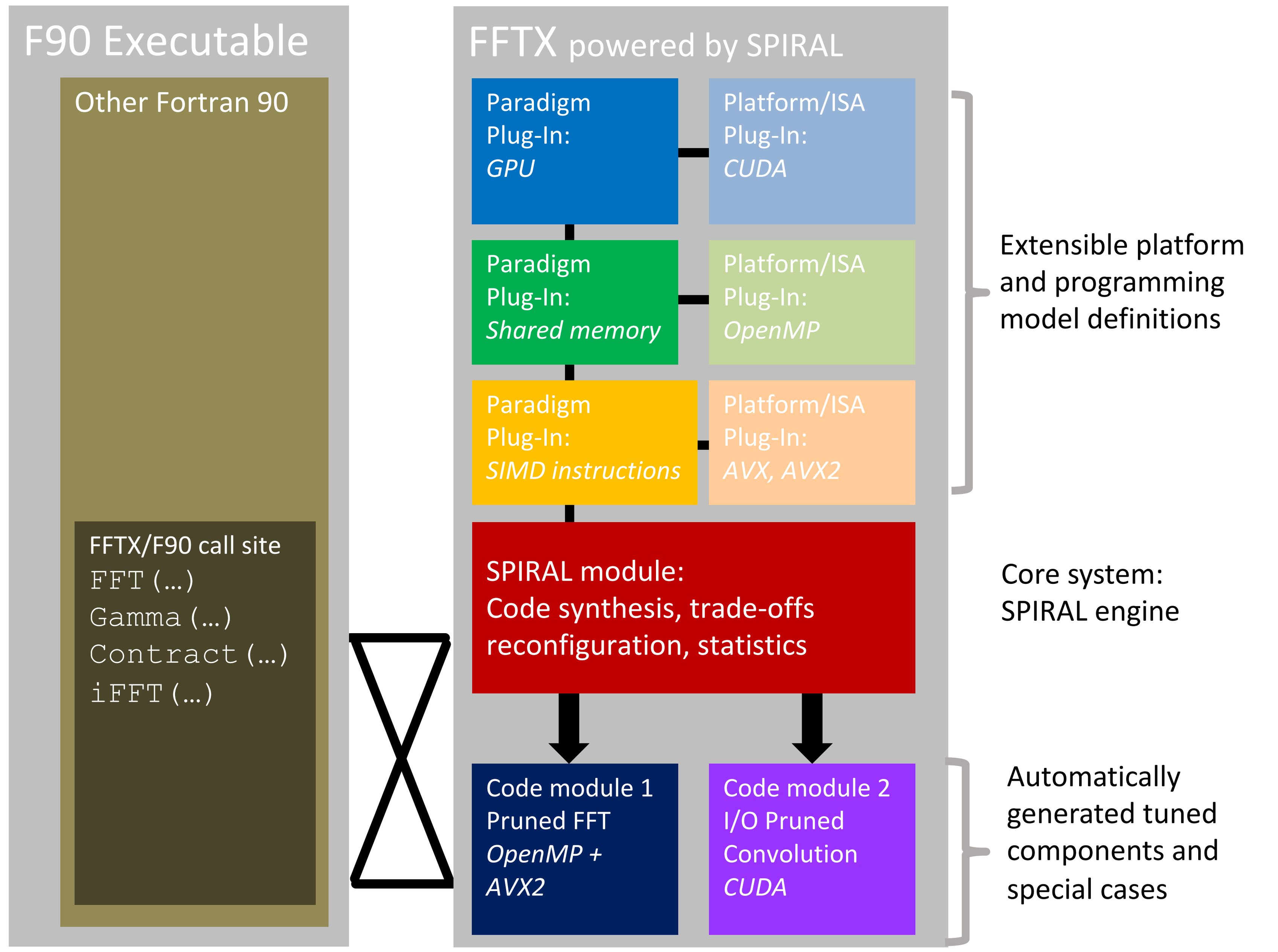
Acknowledgment

This project is supported by DOE ECP project 2.2.6.04 and DOE SciDAC project DE-SC0023523.

Sample EVPFFT code

```
module evpfft_algorithm_mod
    ...
  subroutine outerLoopStep(dt, L0, L0_T4,L0_NEW,vel_grad_correction)
    ...
    ! the fft related Variables
    real(r64) :: K33(3,3),G33(3,3),G33_inv(3,3),Gamma3333(3,3,3,3),&
                 M0(6,6),M0_T4(3,3,3,3)
    ...
    complex(C_DOUBLE_COMPLEX) ::  lambda_fourier_space(3,3), &
        vel_grad_fourier_space(3,3)
    ...
    ! perform forward DFT for the stress field
    call my_fft_data_container%executeForwardDFT()
    ...
    call my_fft_data_container%setPointFromComplexTensor
           (ix,iy,iz, vel_grad_fourier_space)
     ! perform backward DFT
     call my_fft_data_container%executeBackwardDFT()
    ! comnpute the updated velocity gradient
    ...
    ! compute material constituive response
    call elastic_response_augmented_lagrangian(dims_rank, dt, L0,
L0_NEW)
   end subroutine
end module evpfft_algorithm_mod
```

FFTX Backend: SPIRAL



SPIRAL/OL Formalization and Sample Script

$$\hat{\Gamma}^{i,j,k,\ell}_{p,q,r} \circledast \hat{[.]}^{k,\ell}_{p,q,r} \rightarrow \left(\mathrm{iPRDFT}_{N_z\times N_y\times N_x} \otimes \mathrm{I}_{3\times 3}\right) \circ \left(\underline{\Gamma^{i,j,k,\ell}_{p,q,r} : [.]^{k,\ell}_{p,q,r}}\right) \circ \left(\mathrm{PRDFT}_{N_z\times N_y\times N_x} \otimes \mathrm{I}_{3\times 3}\right)$$

$$\underline{\Gamma^{i,j,k,\ell}_{p,q,r} : [.]^{k,\ell}_{p,q,r}} \rightarrow \mathrm{I}_{N_z\times N_y\times N_x/2+1} \otimes_{r,q,p} \left(\overline{\Gamma^{i,j,k,\ell}_{p,q,r}|_{p,q,r} : [.]^{k,\ell}_{p,q,r}|_{p,q,r}}\right)$$

$$\Gamma^{i,j,k,\ell}_{p,q,r}|_{p,q,r} \rightarrow \begin{cases} 0 \in \mathbb{R}^{3\times3\times3\times3}, & \text{if } (p,q,r)=0; \\ -M_0^{i,j,k,\ell}, & \text{if } p=\frac{N_x}{2} \text{ or } q=\frac{N_y}{2} \text{ or } r=\frac{N_z}{2}; \\ \Gamma^{i,j,k,\ell}(\nu^u_{p,q,r}), & \text{else} \end{cases}$$

$$\Gamma^{i,j,k,\ell}(.) \rightarrow \left(\left([.]^\top[.]\right) \otimes \mathrm{I}_{3\times3}\right) \circ \left([.]^{-1} \circ \left(-L_0^{i,j,k,\ell} : [.]\right)[-]\left(\mathrm{I}_{3\times3}\right)\right) \circ \left([.]^\top[.]\right)$$

$$\nu^u_{p,q,r} = 2\pi \begin{bmatrix} \frac{p-N_x\chi_{p\geq N_x/2}}{\Delta_x N_x} \\ \frac{q-N_y\chi_{q\geq N_y/2}}{\Delta_y N_y} \\ \frac{r-N_z\chi_{r\geq N_z/2}}{\Delta_z N_z} \end{bmatrix}$$

```
Load(fftx);
Load(evpfftx);
conf := LocalConfig.fftx.defaultConf();

Nx := 8; Ny := 8; Nz := 8; p := Ind(Nx); q := Ind(Ny); r := Ind(Nz);
i := Ind(3); j := Ind(3); k := Ind(3); l := Ind(3); Nx_ce := Nx + 2; p_ce := Ind(Nx_ce);

lambda_t := TPtr(TTensorField([p, q, r], TTensorValue([i, j], TReal)));
udot_t := TPtr(TTensorField([p, q, r], TTensorValue([k, l], TReal)));

nu0 := var("nu0", TPtr(TTensorValue([p], TReal))); nu1 := var("nu1", TPtr(TTensorValue([q], TReal)));
nu2 := var("nu2", TPtr(TTensorValue([r], TReal)));

M0 := var("M0", TPtr(TTensorValue([i, j, k, l], TReal))); L0 := var("L0", TPtr(TTensorValue([i, j, k, l], TReal)));

Gamma := var("Gamma", TTensorField([p, q, r], TTensorValue([i, j, k, l], TReal)));

lambda_hat := TempVar(TTensorField([p_ce, q, r], TTensorValue([i, j], TReal)));
udot_hat := TempVar(TTensorField([p_ce, q, r], TTensorValue([k, l], TReal)));

t := TFCall(
    TDAG([
        TDAGNode(TRCL(TTensorI(MDPRDFT([Nx, Ny, Nz], -1), 3*3, AVec, AVec), 9), lambda_hat, tcast(lambda_t, X)),
        TDAGNode(TMap(TGammaPoint(nu0, nu1, nu2, L0, M0), [p, q, r], APar, APar), Gamma, [nu0, nu1, nu2, L0, M0]),
        TDAGNode(TContract([Gamma, lambda_hat]), udot_hat, lambda_hat),
        TDAGNode(TRCR(TTensorI(IMDPRDFT([Nx, Ny, Nz], 1), 3*3, AVec, AVec), 9), tcast(udot_t, Y), udot_hat)
    ]),
    rec(fname := name, params := [nu0, nu1, nu2, L0, M0])
);
```

Conclusion & Future Work

- Initial efforts towards generating an optimized EVPFFT code in SPIRAL is shown.
- FFTX is used to improve the FFT computation.
- The future goal is to provide an end-to-end representation in SPIRAL along with performance results on different platforms.

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] F. Franchetti *et al.*, "FFTX and SpectralPack: A First Look," 2018 IEEE 25th International Conference on High Performance Computing Workshops, 2018, pp. 18-27