SPIRAL, FFTX, and the Path to SpectralPACK

Franz Franchetti
Carnegie Mellon University

www.spiral.net

In collaboration with the SPIRAL and FFTX team @ CMU and LBL

This work was supported by DOE ECP and DARPA BRASS
Have You Ever Wondered About This?

Numerical Linear Algebra

- LAPACK
- ScaLAPACK
- LU factorization
- Eigensolves
- SVD
- BLAS, BLACS
  - BLAS-1
  - BLAS-2
  - BLAS-3

Spectral Algorithms

- Convolution
- Correlation
- Upsampling
- Poisson solver
- FFTW
- DFT, RDFT
  - 1D, 2D, 3D,...
  - batch

No LAPACK equivalent for spectral methods

- Medium size 1D FFT (1k—10k data points) is most common library call
  applications break down 3D problems themselves and then call the 1D FFT library

- Higher level FFT calls rarely used
  FFTW guru interface is powerful but hard to used, leading to performance loss

- Low arithmetic intensity and variation of FFT use make library approach hard
  Algorithm specific decompositions and FFT calls intertwined with non-FFT code
It Is Worse Than It Seems

**FFTW is de-facto standard interface for FFT**

- FFTW 3.X is the high performance reference implementation: supports multicore/SMP and MPI, and Cell processor
- Vendor libraries support the FFTW 3.X interface: Intel MKL, IBM ESSL, AMD ACML (end-of-life), Nvidia cuFFT, Cray LibSci/CRAFFT

**Issue 1: 1D FFTW call is standard kernel for many applications**

- Parallel libraries and applications reduce to 1D FFTW call: P3DFFT, QBox, PS/DNS, CPMD, HACC,…
- Supported by modern languages and environments: Python, Matlab,…

**Issue 2: FFTW is slowly becoming obsolete**

- FFTW 2.1.5 (still in use, 1997), FFTW 3 (2004) minor updates since then
- Development currently dormant, except for small bug fixes
- No native support for accelerators (GPUs, Xeon PHI, FPGAs) and SIMT
- Parallel/MPI version does not scale beyond 32 nodes

*Risk: loss of high performance FFT standard library*
**FFTX: The FFTW Revamp for ExaScale**

**Modernized FFTW-style interface**
- Backwards compatible to FFTW 2.X and 3.X
  old code runs unmodified and gains substantially but not fully
- Small number of new features
  futures/delayed execution, offloading, data placement, callback kernels

**Code generation backend using SPIRAL**
- Library/application kernels are interpreted as specifications in DSL
  extract semantics from source code and known library semantics
- Compilation and advanced performance optimization
  cross-call and cross library optimization, accelerator off-loading,…
- Fine control over resource expenditure of optimization
  compile-time, initialization-time, invocation time, optimization resources
- Reference library implementation and bindings to vendor libraries
  library-only reference implementation for ease of development
**FFTX and SpectralPACK: Long Term Vision**

**Numerical Linear Algebra**
- **LAPACK**
  - LU factorization
  - Eigensolves
  - SVD
  - ...
- **BLAS**
  - BLAS-1
  - BLAS-2
  - BLAS-3

**Spectral Algorithms**
- **SpectralPACK**
  - Convolution
  - Correlation
  - Upsampling
  - Poisson solver
  - ...
- **FFTX**
  - DFT, RDFT
  - 1D, 2D, 3D, ...
  - batch

Define the LAPACK equivalent for spectral algorithms

- Define FFTX as the BLAS equivalent
  - provide user FFT functionality as well as algorithm building blocks
- Define class of numerical algorithms to be supported by SpectralPACK
  - PDE solver classes (Green’s function, sparse in normal/k space, ...), signal processing, ...
- Define SpectralPACK functions
  - circular convolutions, NUFFT, Poisson solvers, free space convolution, ...

**FFTX and SpectralPACK solve the “spectral dwarf” long term**
Example: Hockney Free Space Convolution
Example: Hockney Free Space Convolution

```c
fftx_plan pruned_real_convolution_plan(fftx_real *in, fftx_real *out, fftx_complex *symbol,
    int n, int n_in, int n_out, int n_freq) {
    int rank = 1,
    batch_rank = 0,
    ...
    fftx_plan plans[5];
    fftx_plan p;

    tmp1 = fftx_create_zero_temp_real(rank, &padded_dims);
    plans[0] = fftx_plan_guru_copy_real(rank, &in_dimx, in, tmp1, MY_FFTX_MODE_SUB);

    tmp2 = fftx_create_temp_complex(rank, &freq_dims);
    plans[1] = fftx_plan_guru_dft_r2c(rank, &padded_dims, batch_rank,
        &batch_dims, tmp1, tmp2, MY_FFTX_MODE_SUB);

    tmp3 = fftx_create_temp_complex(rank, &freq_dims);
    plans[2] = fftx_plan_guru_pointwise_c2c(rank, &freq_dimx, batch_rank, &batch_dimx,
        tmp2, tmp3, symbol, (fftx_callback)complex_scaling,
        MY_FFTX_MODE_SUB | FFTX_PW_POINTWISE);

    tmp4 = fftx_create_temp_real(rank, &padded_dims);
    plans[3] = fftx_plan_guru_dft_c2r(rank, &padded_dims, batch_rank,
        &batch_dims, tmp3, tmp4, MY_FFTX_MODE_SUB);

    plans[4] = fftx_plan_guru_copy_real(rank, &out_dimx, tmp4, out, MY_FFTX_MODE_SUB);

    p = fftx_plan_compose(numsubplans, plans, MY_FFTX_MODE_TOP);
    return p;
}

Looks like FFTW calls, but is a specification for SPIRAL
```
Spiral Technology in a Nutshell

Library Generator

Traditionally

High performance library optimized for given platform

Spiral

High performance library optimized for given platform

Spiral Approach

Mathematical Foundation

Model: common abstraction = spaces of matching formulas

abstraction

defines

rewriting

search

algorithm space

optimization

Kernel: problem size, algorithm choice

Architectural parameter: Vector length, #processors, ...

Performance Portability

Intel Xeon 8180M
2.25 Tflop/s, 205 W
28 cores, 2.5 — 3.8 GHz
2-way — 16-way AVX/S12

IBM POWER9
768 Gflop/s, 300 W
24 cores, 4 GHz
4-way V/SX-3

Nvidia Tesla V100
7.8 Tflop/s, 300 W
5120 cores, 1.2 GHz
8-way SIMT

Intel Xeon Phi 7290F
1.7 Tflop/s, 260 W
72 cores, 1.5 GHz
8-way/16-way LR8ni

Snapdragon 835
15 Gflop/s, 2 W
8 cores, 2.3 GHz
AS40 GPU, 682 DSP, NEON

Intel Atom C3858
32 Gflop/s, 25 W
16 cores, 2.0 GHz
2-way/4-way SSSE3

Dell PowerEdge R940
3.2 Tflop/s, 647 W
4x 24 cores, 2.1 GHz
4-way/8-way AVX

Summit
187.7 Tflop/s, 13 MW
9,216 x 22 cores POWER9
+ 27,648 V100 GPUs

Code Synthesis and Autotuning

Intel Core i7 (2nd Gen)

Base cases

Transformation rules

Breakdown rules

\( \text{DFT}_{156} \)

\begin{align*}
\text{OL specification} & \rightarrow \text{OL (dataflow)} \rightarrow \Sigma-\text{OL (loop)} \rightarrow \text{Optimized } \Sigma-\text{OL} \\
\text{Confluent term rewriting} & \rightarrow \text{Recursive descent} \rightarrow \text{Optimized abstract code} \\
\text{Recursive descent} & \rightarrow \text{Abstract code} \\
\text{Confluent term rewriting} & \rightarrow \text{Optimized abstract code} \\
\text{Recursive descent} & \rightarrow \text{C code}
\end{align*}
**Algorithms: Rules in Domain Specific Language**

**Linear Transforms**

\[
\begin{align*}
\text{DFT}_n & \rightarrow (\text{DFT}_k \otimes \text{I}_m) T_m^n (\text{I}_k \otimes \text{DFT}_m) \text{L}_k^n, \quad n = km \\
\text{DFT}_n & \rightarrow P_n (\text{DFT}_k \otimes \text{DFT}_m) Q_n, \quad n = km, \quad \text{gcd}(k, m) = 1 \\
\text{DFT}_p & \rightarrow R_p^p (I_1 \otimes \text{DFT}_{p-1}) D_p (I_1 \otimes \text{DFT}_{p-1}) R_p, \quad p \text{ prime} \\
\text{DCT}-3_n & \rightarrow (\text{I}_m \oplus \text{J}_m) L_m^n (\text{DCT}-3_m(1/4) \oplus \text{DCT}-3_m(3/4)) \\
& \cdot (F_2 \otimes I_m) \begin{bmatrix}
1 & 0 \oplus & J_m & -1 \\
1 & \sqrt{2} & (I_1 \oplus 2 I_m)
\end{bmatrix}, \quad n = 2m \\
\text{DCT}-4_n & \rightarrow S_n \text{DCT}-2_n \text{diag}_{0 \leq k < n} (1/(2 \cos((2k + 1)\pi/4n))) \\
\text{IMDCT}_2 & \rightarrow (\text{J}_m \oplus \text{I}_m \otimes \text{I}_m \oplus \text{J}_m) \begin{bmatrix}
1 & 0 \oplus & -1 \\
-1 & 0 \oplus & -1
\end{bmatrix} J_2 \text{DCT}-4_m \\
\text{WHT}_{2k} & \rightarrow \prod_{i=1}^{t} (I_{2k} \oplus \cdots \oplus I_{k}) \otimes \text{WHT}_{2k} \otimes I_{2k+1} \oplus \cdots \oplus I_t, \quad k = k_1 + \cdots + k_t \\
\text{DFT}_2 & \rightarrow F_2 \\
\text{DCT}-2 & \rightarrow \text{diag}(1, 1/\sqrt{2}) F_2 \\
\text{DCT}-4 & \rightarrow J_2 R_{13\pi/8}
\end{align*}
\]

**Graph Algorithms**

**Numerical Linear Algebra**

\[
\begin{align*}
\text{MM} & \rightarrow (\cdot)_1 \\
\text{MM} & \rightarrow (\otimes)_{m \times m} \otimes \text{MM}_{m \times m} \\
\text{MM} & \rightarrow \text{MM} \otimes (\otimes)_{m \times m} \\
\text{MM} & \rightarrow (\Sigma_{k} \otimes (\cdot)_{k}) \otimes \text{MM}_{m \times m} \\
\text{MM} & \rightarrow (L_{m} \otimes I_{m}) \otimes (L_{m} \otimes I_{m}) \\
\text{MM} & \rightarrow (L_{m} \otimes I_{m}) \otimes (L_{m} \otimes I_{m})
\end{align*}
\]

**Spectral Domain Applications**

- preprocessing
- matched filtering
- interpolation
- 2D iFFT

---

**In collaboration with CMU-SEI**
SPIRAL: Success in HPC/Supercomputing

- NCSA Blue Waters
  PAID Program, FFTs for Blue Waters

- RIKEN K computer
  FFTs for the HPC-ACE ISA

- LANL RoadRunner
  FFTs for the Cell processor

- PSC/XSEDE Bridges
  Large size FFTs

- LLNL BlueGene/L and P
  FFTW for BlueGene/L’s Double FPU

- ANL BlueGene/Q Mira
  Early Science Program, FFTW for BGQ QPX

Global FFT (1D FFT, HPC Challenge) performance [Gflop/s]

6.4 Tflop/s on BlueGene/P at Argonne National Laboratory
128k cores (quad-core CPUs) at 850 MHz

2006 Gordon Bell Prize (Peak Performance Award) with LLNL and IBM
2010 HPC Challenge Class II Award (Most Productive System) with ANL and IBM
**FFTX Backend: SPIRAL**

**Executable**

- Other C/C++ Code

**FFTX call site**

- `fftx_plan(...)`
- `fftx_execute(....)`

**FFTX powered by SPIRAL**

- Paradigm Plug-In: GPU
- Paradigm Plug-In: Shared memory

**SPIRAL module:**

- Code synthesis, trade-offs, reconfiguration, statistics

**FFTW-like library components**

- Core system: SPIRAL engine
- Automatically Generated
- FFT Codelets `CUDA`
- FFT Solvers `OpenMP`

**Extensible platform and programming model definitions**

DARPA BRASS
FFTX: First Results for Hockney on Volta

FFTX with SPIRAL and OpenACC:
on par with cuFFT expert interface

FFTX with SPIRAL and OpenACC:
15 % faster than cuFFT expert interface


http://www.spiral.net/doc/fftx
SPIRAL 8.0: Available Under Open Source

- Open Source SPIRAL available
  - non-viral license (BSD)
  - Initial version, effort ongoing to open source whole system
  - Commercial support via SpiralGen, Inc.

- Developed over 20 years
  - Funding: DARPA (OPAL, DESA, HACMS, PERFECT, BRASS), NSF, ONR, DoD HPC, JPL, DOE, CMU SEI, Intel, Nvidia, Mercury

- Open sourced under DARPA PERFECT

www.spiral.net

Special Issue on From High-Level Specification to High Performance Code