Spiral: Specialized FFTs

At ESSL and FFTW Speed

Franz Franchetti

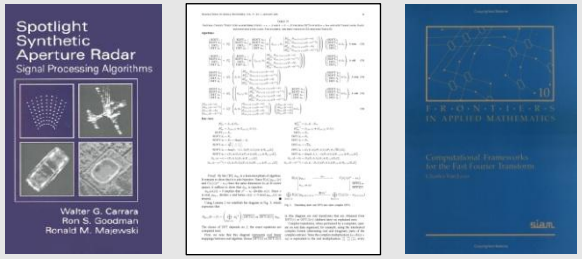
Carnegie Mellon University
`www.ece.cmu.edu/~franzf`

CTO and Co-Founder, SpiralGen
`www.spiralgen.com`

This work was supported by
DARPA DESA program, NSF, ONR, Mercury Inc., Intel, and Nvidia

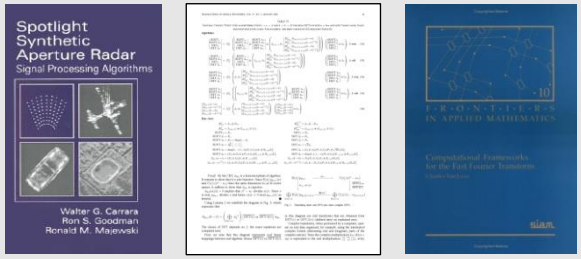
What is Spiral?

Traditionally



High performance library optimized for given platform

Spiral Approach



Spiral

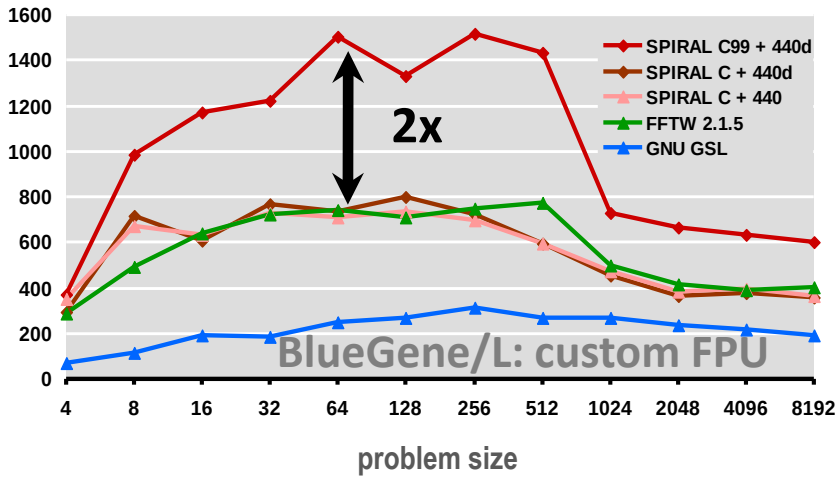
High performance library optimized for given platform

Comparable performance

BlueGene/L and P Node Performance

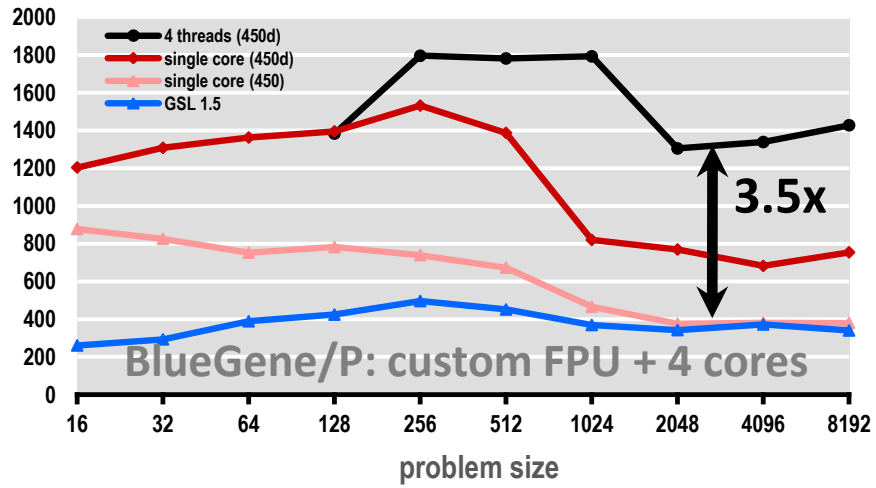
DFT, double precision, XL C compiler

performance [Mflop/s]



DFT, double precision, XL C compiler

performance [Mflop/s]



Single BlueGene/L CPU at 700 MHz
IBM T. J. Watson Research Center

SIMD vectorization

Single BlueGene/P node (4 CPUs) at 850 MHz
Argonne National Laboratory

SIMD vectorization + multi-threading

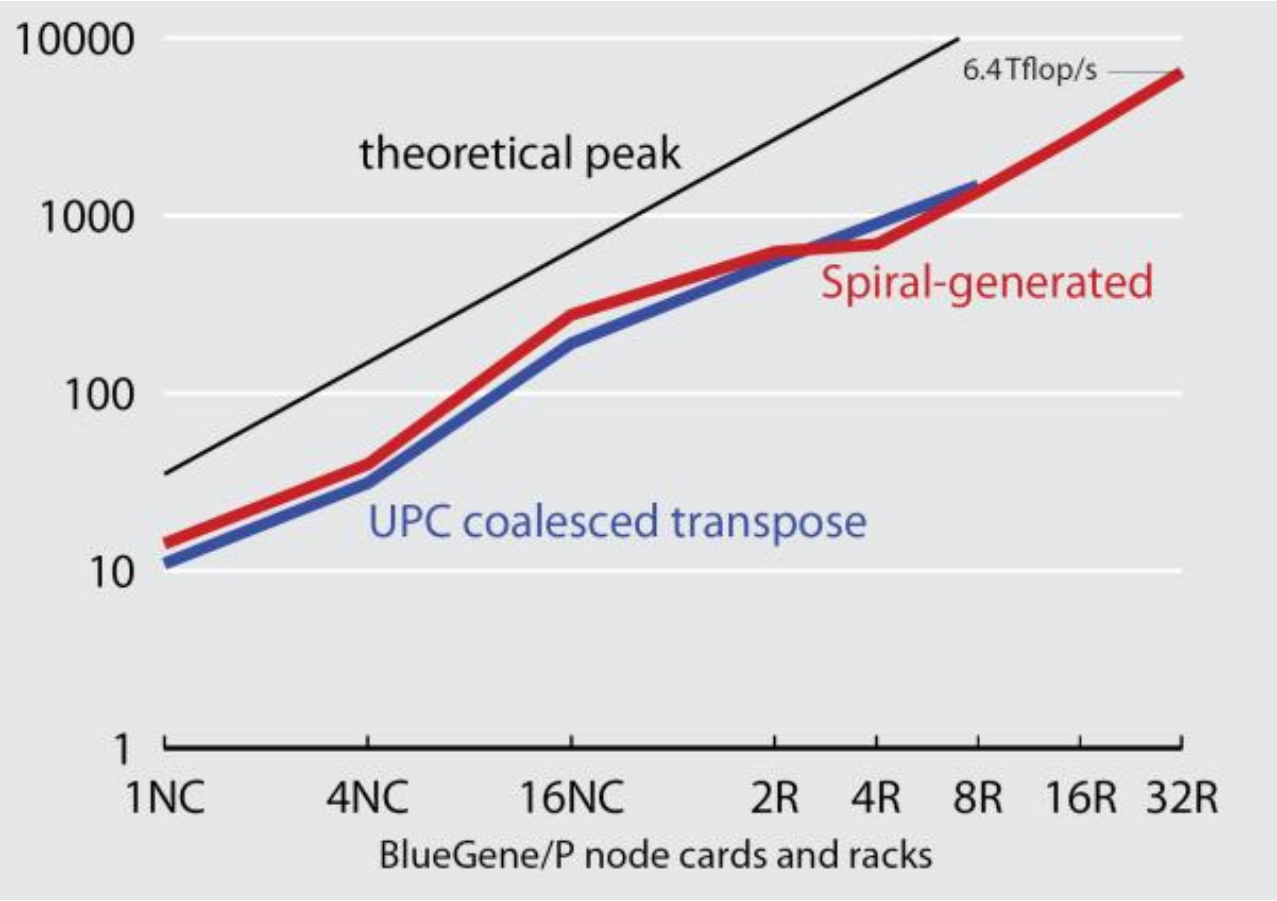
F. Gygi, E. W. Draeger, M. Schulz, B. R. de Supinski, J. A. Gunnels, V. Austel, J. C. Sexton, F. Franchetti, S. Kral, C. W. Ueberhuber, J. Lorenz: **Large-Scale Electronic Structure Calculations of High-Z Metals on the BlueGene/L Platform.** In Proceedings of Supercomputing, 2006. **Winner of the 2006 Gordon Bell Prize (Peak Performance Award).**

J. Lorenz, S. Kral, F. Franchetti, C. W. Ueberhuber: **Vectorization Techniques for the Blue Gene/L double FPU.** IBM Journal of Research and Development, Vol. 49, No. 2/3, 2005.

HPC Challenge Global FFT on BlueGene/P

1D Global FFT

performance [Gflop/s]

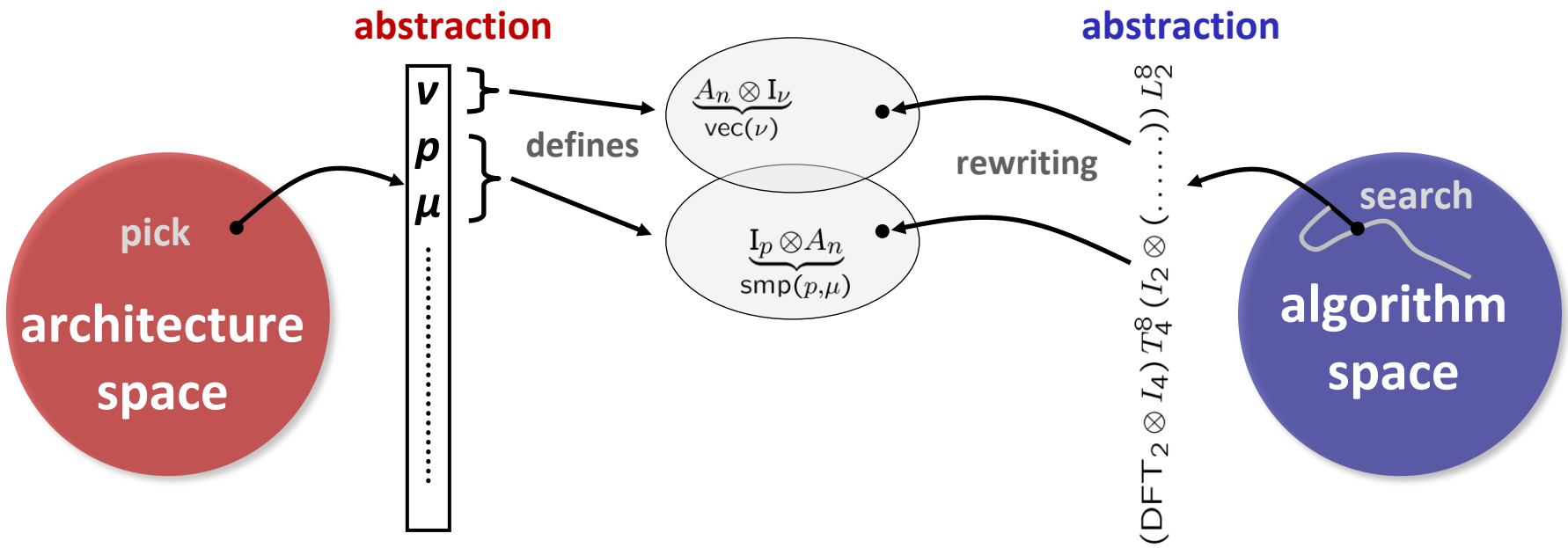


6.4 Tflop/s

G. Almási, B. Dalton, L. L. Hu, F. Franchetti, Y. Liu, A. Sidelnik, T. Spelce, I. G. Tānase, E. Tiotto, Y. Voronenko, X. Xue:
2010 IBM HPC Challenge Class II Submission. Winner of the 2010 HPC Challenge Class II Award (Most Productive System).

Spiral's Domain-Specific Program Synthesis

Model: common abstraction
= spaces of matching formulas



Architectural parameter:
Vector length,
#processors, …

optimization

Kernel:
problem size,
algorithm choice

Spiral's Formal Framework

- **Transform = Matrix-vector multiplication**

  Example: Discrete Fourier transform (DFT)

$$x \mapsto y = T \cdot x$$

input vector (signal)

output vector (signal)

transform = matrix

- **Fast algorithm = sparse matrix factorization = SPL formula**

  Example: Cooley-Tukey FFT algorithm

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} = \begin{bmatrix} 1 & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & 1 \\ 1 & \cdot & -1 & \cdot \\ \cdot & 1 & \cdot & -1 \end{bmatrix} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & j \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdot & \cdot \\ 1 & -1 & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 \end{bmatrix}$$

$$\mathrm{DFT}_4 = (\mathrm{DFT}_2 \otimes \mathrm{I}_2)\,\mathsf{T}^4_2\,(\mathrm{I}_2 \otimes \mathrm{DFT}_2)\,\mathsf{L}^4_2$$

Transforms and Breakdown Rules

$$\mathbf{DFT}_n \rightarrow (\mathbf{DFT}_k \otimes \mathrm{I}_m)\, \mathsf{T}^n_m (\mathrm{I}_k \otimes \mathbf{DFT}_m)\, \mathsf{L}^n_k, \quad n = km$$

$$\mathbf{DFT}_n \rightarrow P_n (\mathbf{DFT}_k \otimes \mathbf{DFT}_m) Q_n, \quad n = km,\ \gcd(k,m) = 1$$

$$\mathbf{DFT}_p \rightarrow R_p^T (\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1}) D_p (\mathrm{I}_1 \oplus \mathbf{DFT}_{p-1}) R_p, \quad p \text{ prime}$$

DCT-3

DCT-4

"Teaches" Spiral about existing algorithm knowledge (~200 journal papers)

$$\mathbf{IMDCT}_{2m} \rightarrow (\mathrm{J}_m \oplus \mathrm{I}_m \oplus \mathrm{I}_m \oplus \mathrm{J}_m) \left(\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \otimes \mathrm{I}_m \right) \oplus \left(\begin{bmatrix} -1 \\ -1 \end{bmatrix} \otimes \mathrm{I}_m \right) \right) \mathrm{J}_{2m}\, \mathbf{DCT\text{-}4}_{2m}$$

$$\mathbf{WHT}_{2^k} \rightarrow \prod_{i=1}^{t} (\mathrm{I}_{2^{k_1+\cdots+k_{i-1}}} \otimes \mathbf{WHT}_{2^{k_i}} \otimes \mathrm{I}_{2^{k_{i+1}+\cdots+k_t}}), \quad k = k_1 + \cdots + k_t$$

$$\mathbf{DFT}_2 \rightarrow \mathsf{F}_2$$

$$\mathbf{DCT\text{-}2}_2 \rightarrow \mathrm{diag}(1, 1/\sqrt{2})\, \mathsf{F}_2$$

$$\mathbf{DCT\text{-}4}_2 \rightarrow \mathrm{J}_2\, \mathsf{R}_{13\pi/8}$$

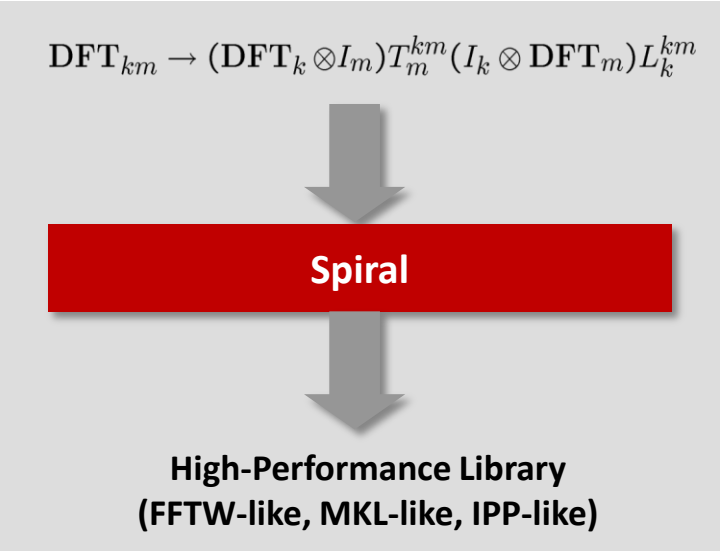
Base case rules

Teaches Spiral about FFT algorithms

Synthesizing FFTW From Its Specification

Input:

- **Transform**: DFT_n
- **Algorithms**: $\mathrm{DFT}_{km} \rightarrow (\mathrm{DFT}_k \otimes I_m) T_m^{km} (I_k \otimes \mathrm{DFT}_m) L_k^{km}$
  $\mathrm{DFT}_2 \rightarrow \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- **Vectorization**: 2-way SSE
- **Threading**: Yes



Output:

- Optimized library (10,000 lines of C++)
- For general input size
  (**not** collection of fixed sizes)
- Vectorized
- Multithreaded
- With runtime adaptation mechanism
- Performance competitive with hand-written code

Example Specialized FFTs Pruned FFT

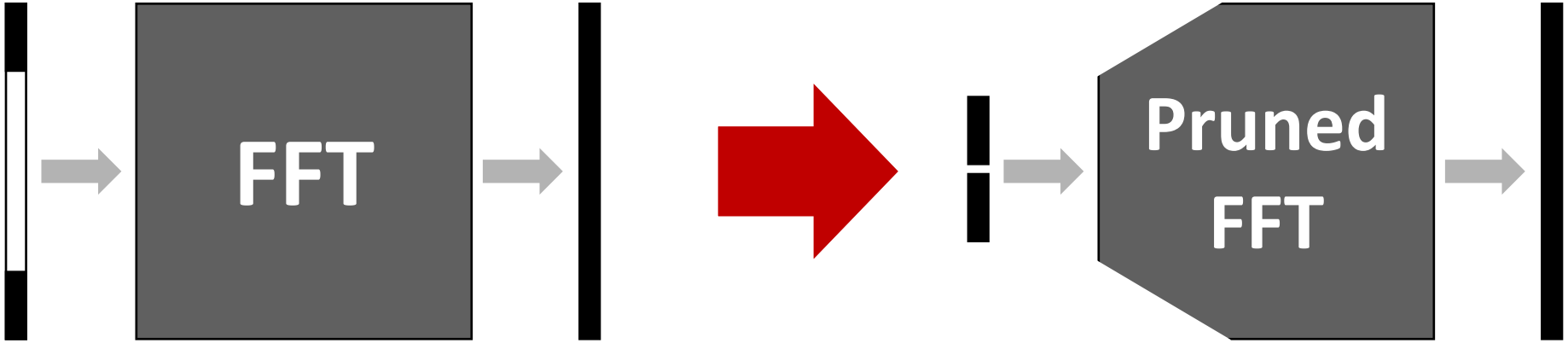
- **Recursive input pruning rule**

$$\mathrm{PDFT}_{kmn}^{\sigma \otimes km} \rightarrow (\mathrm{DFT}_m \otimes I_{kn}) T_{kn}^{kmn} L_m^{kmn} (\mathrm{PDFT}_{kn}^{\sigma \otimes k} \otimes I_m)$$

- **Base case**

$$\mathrm{PDFT}_n^{\sigma} \rightarrow \mathrm{DFT}_n \mathrm{S}_{\sigma}$$

- **Similar rule for output pruning and simultaneous pruning**



Spiral And BlueGene/Q

- **Single node library**
  fully optimized FFTW replacement

- **Multi-node replacement**
  generate optimized P3DFFT

- **Specialized FFTs for applications**
  work with HACC and PS/DNS

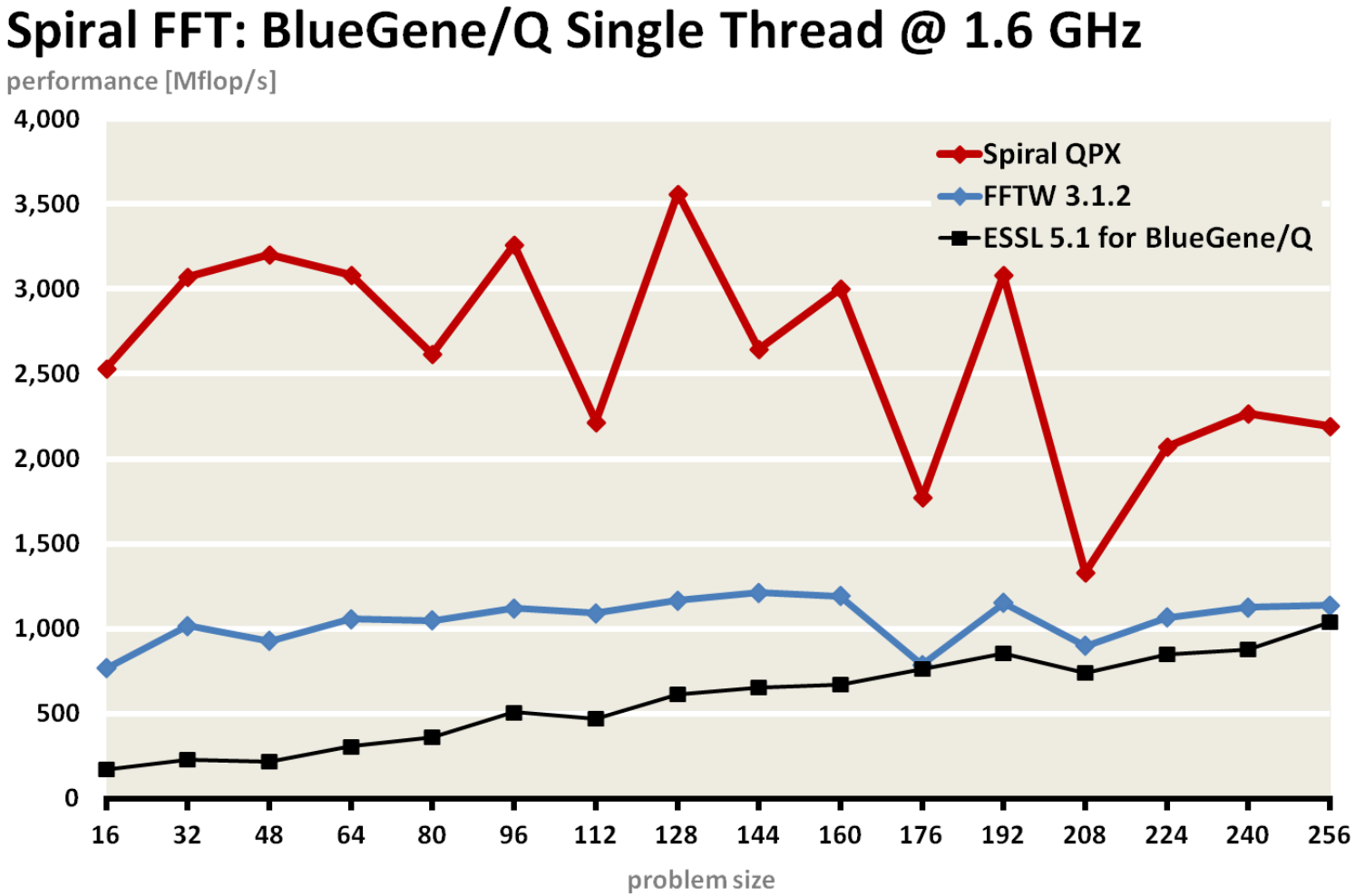
- **Higher-level FFT-heavy kernels**
  convolution, correlation, Poisson solvers

The Fun Just Started…

```
void dft16(double  *Y, double  *X) {
    const vector4double C1 = (vector4double)(1.0, 0.70710678118654757, 0.0, (-0.70710678118654757));
    const vector4double C2 = (vector4double)(0.0, 0.70710678118654757, 1.0, 0.70710678118654757);
    const vector4double C3 = (vector4double)(1.0, 0.92387953251128674, 0.70710678118654757, 0.38268343236508978);
    const vector4double C4 = (vector4double)(0.0, 0.38268343236508978, 0.70710678118654757, 0.92387953251128674);
    const vector4double C5 = (vector4double)(1.0, 0.38268343236508978, (-0.70710678118654757), (-0.92387953251128674));
    const vector4double C6 = (vector4double)(0.0, 0.92387953251128674, 0.70710678118654757, (-0.38268343236508978));
    vector4double a90, a91, a92, a93, a94, a95, s139, s140, s141, s142, s143, s144, s145, s146, s147, s148,...,;
    vector4double  *a89, *a96;
    a89 = ((vector4double  *) X);
    s139 = a89[0];
    s140 = a89[1];
    a90 = vec_gpci(0xa60);
    s141 = vec_perm(s139, s140, a90);
    a91 = vec_gpci(0xef2);
    s142 = vec_perm(s139, s140, a91);
    s143 = a89[4];
    s144 = a89[5];
    s145 = vec_perm(s143, s144, a90);
    ...
    s170 = vec_perm(s158, s162, a95);
    s171 = vec_sub(vec_mul(C1, s165), vec_mul(
    s172 = vec_add(vec_mul(C2, s165), vec_mul(
    t145 = vec_add(s163, s171);
    t146 = vec_add(s167, s172);
    t147 = vec_sub(s163, s171);
    t148 = vec_sub(s167, s172);
    s173 = vec_sub(vec_mul(C3, s164), vec_mul(
    s174 = vec_add(vec_mul(C4, s164), vec_mul(
    s175 = vec_sub(vec_mul(C5, s166), vec_mul(
    s176 = vec_add(vec_mul(C6, s166), vec_mul(
    t149 = vec_add(s173, s175);
    ...
    a96[3] = s182;
    s183 = vec_perm(t159, t160, a92);
    a96[6] = s183;
    s184 = vec_perm(t159, t160, a93);
    a96[7] = s184;
}
```

```
78| 00014C qvfmul   118D0132   1     QVFMUL    qr12=qr13,qr4,fcr
79| 000150 qvfmul   11AD0172   1     QVFMUL    qr13=qr13,qr5,fcr
84| 000154 qvfmul   11CF01B2   1     QVFMUL    qr14=qr15,qr6,fcr
85| 000158 qvfmul   11EF01F2   1     QVFMUL    qr15=qr15,qr7,fcr
86| 00015C qvfmul   12110232   1     QVFMUL    qr16=qr17,qr8,fcr
87| 000160 qvfmul   12310272   1     QVFMUL    qr17=qr17,qr9,fcr
60| 000164 qvfperm  1253B00C   1     QVFPERM   qr18=qr19,qr22,qr0
62| 000168 qvfperm  1273B04C   1     QVFPERM   qr19=qr19,qr22,qr1
65| 00016C qvfperm  12F4A80C   1     QVFPERM   qr23=qr20,qr21,qr0
66| 000170 qvfperm  1294A84C   1     QVFPERM   qr20=qr20,qr21,qr1
72| 000174 qvfperm  12B2B8CC   1     QVFPERM   qr21=qr18,qr23,qr3
73| 000178 qvfperm  12D3A08C   1     QVFPERM   qr22=qr19,qr20,qr2
74| 00017C qvfperm  1073A0CC   1     QVFPERM   qr3=qr19,qr20,qr3
79| 000180 qvfnmadd 10B62B3E   1     QVFNMADD  qr5=qr5,qr22,qr12,fcr
80| 000184 qvfmadd  1096237A   1     QVFMADD   qr4=qr4,qr22,qr13,fcr
85| 000188 qvfnmadd 10F53BBE   1     QVFNMADD  qr7=qr7,qr21,qr14,fcr
86| 00018C qvfmadd  10D533FA   1     QVFMADD   qr6=qr6,qr21,qr15,fcr
87| 000190 qvfnmadd 11234C3E   1     QVFNMADD  qr9=qr9,qr3,qr16,fcr
88| 000194 qvfmadd  1063447A   1     QVFMADD   qr3=qr8,qr3,qr17,fcr
70| 000198 qvfperm  1112B88C   1     QVFPERM   qr8=qr18,qr23,qr2
75| 00019C qvfperm  104A588C   1     QVFPERM   qr2=qr10,qr11,qr2
81| 0001A0 qvfadd   1148282A   1     QVFADD    qr10=qr8,qr5,fcr
82| 0001A4 qvfadd   1162202A   1     QVFADD    qr11=qr2,qr4,fcr
89| 0001A8 qvfadd   1187482A   1     QVFADD    qr12=qr7,qr9,fcr
90| 0001AC qvfadd   11A6182A   1     QVFADD    qr13=qr6,qr3,fcr
83| 0001B0 qvfsub   10A82828   1     QVFSUB    qr5=qr8,qr5,fcr
```

First Performance Results



More Information:

www.spiral.net

www.spiralgen.com