



VSIPL++: Parallel VSIPL Using C++

CodeSourcery, LLC

November 17, 2002

VSIPL++ Objectives

- Productivity
 - High-level abstractions
- Portability
 - Standard ISO C++, MPI
- Performance
 - Proven techniques from POOMA
- Parallelism
 - SPMD

Development Benefits

- Automatic parallelism
 - No explicit MPI calls
- Code uses mathematical notation
- Automatic memory allocation

- Fewer opportunities for error
- Reduced development costs
- Reduced development time

Functional Objectives

- Improve performance:
 - Parallelism
 - Reduce memory use
 - Enable loop fusion
- Provide extensible interface:
 - Optimized data storage formats
 - Automatic parallelization of new operators
- Simplify client code:
 - Fewer opportunities for error.

k-Omega Beamforming Program

- VSIPL k- Ω beamforming program
 - Transformed to VSIPL++
- Productivity improvement:
 - Intuitive syntax
 - 2.5x fewer lines than VSIPL version
- Performance improvement:
 - Avoid explicit temporary matrices and vectors

k-Omega Beamforming Code

```
Matrix<double> m_data(...);  
Matrix<double> m_gram (...);  
FFT<Matrix, double,  
    complex<double>, ...>  
    time2freq(...);  
FFT<Matrix, complex<double>,  
    complex<double>, ...>  
    freq2dir(...);
```

k-Omega Inner Loop

```
m_gram += 1.0/num_avgs *  
magsq(freq2dir(time2freq(  
vmmul<COL>(hanning_sensors,  
vmmul<ROW>(hanning_tsl,  
m_data)))));
```

k-Omega Inner Loop

- Inner loop
 - Expressed in one line
 - Direct implementation of math specification
- Uses C++ tools:
 - Function objects: FFT application
 - Template parameters
 - Functions return vectors

Blocks and Views

- **Blocks**
 - How data is stored and computed
- **Views**
 - How data is used
 - Vectors, Matrixes
 - Support data-parallel computation
 - Supports distributed computation

Fast, Data-Parallel Expressions

- Support data-parallel operations

```
v0 = v1 + v2 * v3 + 7*v4
```

- Math-like syntax
- Possible implementation:
 - Uses one outermost loop
 - Makes extensive use of C++ templates

BLAS zherk Routine

- BLAS = Basic Linear Algebra Subprograms
- Hermitian matrix M : $\text{conjug}(M) = M^t$
- zherk performs a rank-k update of Hermitian matrix C :

$$C \leftarrow \alpha * A * \text{conjug}(A)^t + \beta * C$$

VSIPL++ Version

```
Matrix<complex<double> > A(10,15);  
Matrix<complex<double> > C(10,10);  
// No need for "tmp" matrix.  
  
C = alpha * prodh(A,A) + beta * C;  
  
// Matrices and blocks automatically destroyed.
```

VSIPL++ Realization

```
const length domSz = A.domain()[1].get_length();
for (index i = 0 ...) { // loop over rows
    for (index j = 0 ...) { // loop over columns
        complex<double> prod = 0.0;
        for (index idx = 0; idx < domSz; ++idx)
            prod += cjmul(A.get(i,idx), A.get(j,idx));
        C.put(i,j) = alpha*prod + beta*C.get(i,j);
    }
}
```

VSIPL++ Advantages

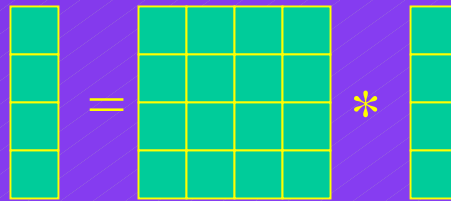
- Storage reduction:
 - Elimination of temporary matrix
- Loop fusion:
 - Reduced computation
- Simpler code:
 - Fewer lines
 - More intuitive syntax

Parallel Computation

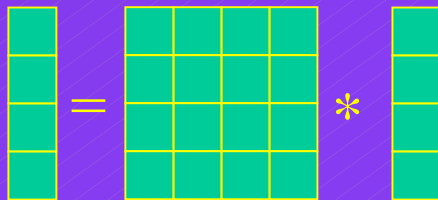
- Conceptual model:
 - SPMD
 - Computation occurs near data
- Implementations:
 - Threads
 - MPI
- Supports multiple hardware/software interfaces.

Matrix Vector Multiplication

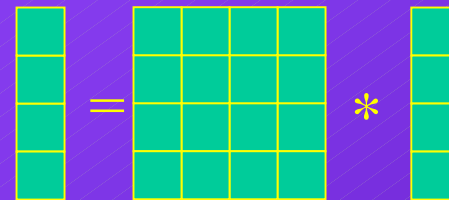
Serial



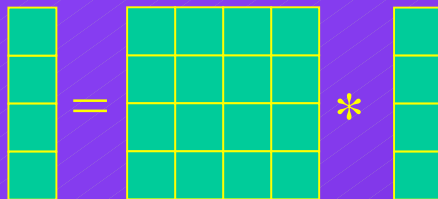
P1



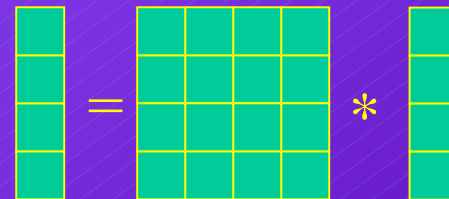
P3



P2



P4



Data Distribution

- Map rectangular regions of a block to zero or more processors:
 - Zero processors: unstored data
 - One processor: distributed data
 - Multiple processors: replicated data
- Read distribution:
 - How data is distributed for reading
- Write distribution:
 - How data is distributed for writing

Parallel zherk

- Specify and use a distribution \bar{d} :

```
Distribution d(UniformPartition<2>(1,
    min(20, getNumberOfProcessors())));
Matrix<complex<double> > A(d, 10, 15);
Matrix<complex<double> > C(d, 10, 10);
C = alpha * prodh(A, A) + beta * C;
```

Contact Information

- Mark Mitchell

mark@codesourcery.com

- Jeffrey Oldham

oldham@codesourcery.com



VSIPL++: Parallel VSIPL Using C++

CodeSourcery, LLC

November 17, 2002