

# **Experiences in Porting an Existing Application to use the VSIP API**

**Kenneth Cain Jr.**

**Brian Sroka**

**July 22, 1997**

**Organization: D730  
Project: 1597 2589 80**

**MITRE**

## Application: RT\_STAP

- **A Real-Time Space-Time Adaptive Processing Benchmark**
- **A report describing the benchmark is available at:  
<http://www.mitre.org/research/hpc>**
- **MATLAB and sequential C implementation of the benchmark is also available (see report)**
- **Three post-Doppler space-time adaptive processing algorithms:**
  - **Displaced Phase Center Antenna (DPCA)**
  - **First-Order Doppler Factored STAP**
  - **Third-Order Doppler Factored STAP**
- **Single or double precision floating point**

# RT\_STAP Software Characteristics

- **Custom vector operations API**
  - **Abstraction layer to support multiple implementations**
    - **Straight C, Mercury SAL, Sun Sunsoft**
  - **To be replaced by VSIP API**
- **Custom floating point complex data type**
  - **Custom API for performing complex number operations**
  - **Support either single or double precision floating point**
  - **Implemented using function inlining for performance**
- **Data shapes: vectors, matrices, and cubes**
- **Multidimensional objects (matrices, cubes) are generated with intermediate pointers to allow a (natural) array-indexing access method**

## VSIP Reference Library Used

- Downloaded version dated 07/03/97 from [www.vsip.org](http://www.vsip.org)
- Built successfully on SunOS 5.5, IRIX 5.3, IRIX 6.4, LINUX
- Generated additional set of classes:
  - Single-precision floating-point scalars, blocks, and vector views
  - Short integer scalars, blocks, and vector views
  - Complex single-precision floating-point scalars and vector views

# VSIP Data Representation Issues

- **Object-oriented API must be injected into a program that is inherently procedural**
  - In an existing program, C array indexing / pointers are the preeminent array referencing mechanisms
    - **Example:  $x = v[i]$**
  - In an OO program, accessor functions should be used to reference all data items
    - **Example:  $x = vget(v, i)$**
- **Higher dimensional objects (matrices, tensors) not yet implemented by VSIP**
  - API for matrices exist but no reference implementation
  - No VSIP API for data cubes

# Partial Solutions to Representation Issues for Existing Source Code

- **To handle the object-oriented style problem:**
  - All vector views are bound to public data vectors
  - This allows existing array indexing / pointer code to still function properly
  - This should eventually be phased out!
  - Has performance implications today
  - Transition tool
- **To implement matrices and cubes:**
  - We implement higher dimensional objects as an array of vector view objects
    - Example: a cube of depth  $D$ , height  $H$ , and width  $W$  is created as  $D \times H$  VSIP vector views (each referring to a vector of length  $W$ )
    - Why: need to process faces and vectors

# Current Port Status

- **3 staff weeks**
  - RT\_STAP designer and OO programmer (MS, 4 years)
  - OO programmer (BS, new hire)
- **Data type conversion performed for every critical vector, matrix, and datacube used in the software**
  - Public arrays bound to VSIP vector views, for now
  - Not easily done in a “catch all” fashion
    - Follow around changed variables
- **Macro replacement of data types and function call names to seamlessly support both single- and double-precision floating-point values (both real and complex)**
- **Some time spent optimizing current performance**

# Lines of Code

- **RT\_STAP separated into application and library parts--only consider application part here**
  - **RT\_STAP before: 3259 semicolons**
    - **Variable/data type definitions/declarations: 15%**
    - **Function bodies: 75%**
    - **Function/library calls: 10%**
- **Total lines of code increased by 27% with backward compatibility**
  - **RT\_STAP after: 4135 semicolons**
  - **VSIP code expansion resulted because of new data types and supporting functions**
    - **Example: new memory allocations using VSIP public vector views**
- **5% increase if backward compatibility not maintained**

## Conclusion (1 of 2)

- **Public arrays were essential as a transition tool**
  - Focus attention on transitioning legacy code issues
- **Compile-time work around for multiple precision modes**
  - Function overloading as in C++ for run-time
- **Moderate code expansion suggests code bloat won't be an issue**
- **Performance penalty concern**
  - Library calls (FFTs, inner products,...)
    - Vendors better provide optimized versions
  - Other (specialized algorithms, I/O, conditional data flows, ...)
    - Better OO compilers will be needed
    - Penalty of up to a factor of 2 today

## Conclusion (2 of 2)

- **Key issue: what percentage of run-time is taken up by non-library code?**
  - **In RT\_STAP 80% of the complexity is in the QR-decomposition**
    - **Will this be a VSIP function call?**
- **Multiprocessing issues/concerns:**
  - **Thread safe VSIP calls**
  - **Data parallel OO implementations**
    - **Encapsulation hides data representation**
  - **Performance is key**