NRC·CNRC Institute for Biodiagnostics

La Science à l'œuvre pour le Canada / at work for Canada

Scopira: An Open Source C++ Framework for Biomedical Data Analysis Applications – A Research Project Report

Aleksander Demko[1,2], Rodrigo A. Vivanco[1], Nick J. Pizzi[1,2]

[1]Institute for Biodiagnostics, National Research Council of Canada, Winnipeg, Canada. [2]Department of Computer Science, University of Manitoba, Winnipeg, Canada

Scopira

Scopira is an open source, object-oriented and generic C++ framework for scientific computing applications with emphasis on biomedical data analysis.

- Spectra and images (magnetic resonance, infrared, gene microarrays, Raman, mass spec).
- Visualization (2D and 3D via OpenGL).
- Computationally efficient.
- Parallel algorithm development (MPI and a dedicated *agent* facility).

Foundation

tool Subsystem

The *tool* subsystem provides a host of generic facilities and utilities for all types of Scopira based applications.

- Reference counting with "smart pointers".
- Threads and concurrent programming.
- Random number generators ("real" and pseudo) and distributions.
- Input/output & object serialization.

I/O & Serialization: *flows*

Scopira contains a layered I/O system with *flows*. Flows provide a three tier interface architecture for the I/O of bytes, simple data types and objects (serialization).

Binary sinks (end terminators for binary streams)
Network, Disk files, Memory

Binary transformers (transform binary streams)
Hex encoding, Encryption, Compression, etc

Data type encoding layer (converts basic data types to bytes)
Convert to ASCII, Convert to Binary

Object serialization layer (converts objects to data types)

core Subsystem

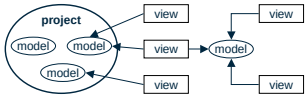
The *core* subsystem builds on the *tool* subsystem and provides non-graphical features useful for building applications.

- **basic_loop** main loop for parsing configuration options.
- Powerful plug-in loading systems.
- Flexible object registration system for registering objects for serialization and virtual construction.
- Dynamic *model*/*view* system.

Models and Views

Scopira provides a dynamic system (via run-time registration) for applications and plug-ins to declare *models* and *views*.

- *Models* are objects that are monitored by zero or more *views*. Views themselves can be graphical or non-graphical.
- A *project* is a model that organizes a collection of models in a tree like fashion. Users may use projects to save their data sets as related workspaces.

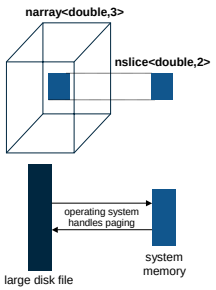


basekit Subsystem

The *basekit* subsystem contains various numerical routines and the core numerical data structure, **narray**.

- Generic, via C++ templates structure. Can be used with ANY data types (ints, floats, complex, etc) and ANY dimension (vector, matrix, cube, etc).
- Complementary **nslice** virtual sub-view, any size <= **narray** dimensions.
- As-good-as-C performance via templates and inline methods.
- Range checked access via **assert()**, (debug mode only).
- STL-style iterators and thereby usable with STL algorithms.
- **DirectIO** back end, to directly access files as if they were in memory (via the operating system's *mmap* function)

DirectIO: Memory Mapping



Scopira Framework

Foundation	GTK+ Graphics
tool	*coreui*
core	*uikit*
basekit	*lab*
Parallel Computing	**OpenGL**
mpi	*coregl*
agent	

Parallel Computing

mpi

This subsystem provides a light (inline) helper layer for Scopira apps that use the MPI (Message Passing Interface) API.

- Aware of **narrays** and uses type information (via C++ traits classes) to deduce many typical parameters needed for many MPI functions.
- Drastically reduce the amount of information needed from the programmer.

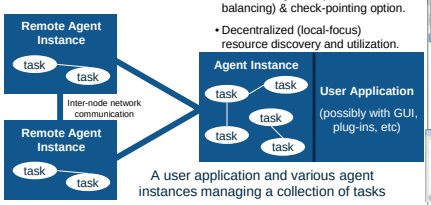
agent

Agent based user application include:

- Agent objects, which represents the application on an agent network.
- *Tasks* (agent-managed) that send *messages* to other tasks & migrate between agents over the network.

This subsystem will include:

- Parallel algorithm API. Focus: OOD, **narrays**, & flow-based serialization.
- Flow-based messaging API.
- Efficient, scalable message routing and task migration (cf. load balancing) & check-pointing option.
- Decentralized (local-focus) resource discovery and utilization.



A user application and various agent instances managing a collection of tasks

agent types

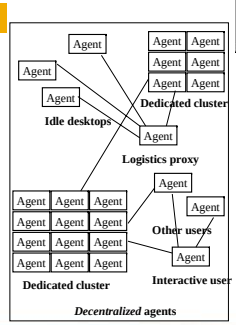
Local: non-network aware, single machine (but multi-threaded).

Cluster (under development): Used with dedicated, fully connected and persistent Beowulf-like clusters.

- Load balancing done at global level as resource allocation decisions.

Decentralized (under development): For larger, complex agent networks.

- Allocate resources based on local information only (permits network scalability); may be used over unstable network links to possibly unreliable remote agents.
- Most dynamic agent, requiring many peer-to-peer like approaches to resource allocation/deployment.



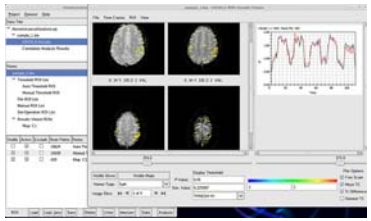
Applications

Scopira used to build a suite of applications and plug-ins, each with varying scopes of complexity:

- Algorithms prototyping, possibly with visualization (*BigVol*)
- Parallelized algorithms, via MPI, for the analysis and classification of biomedical data (*SFS*).
- Plug-ins for pattern recognition algorithms, visualization, and data projection (*RDP*).
- Full, stand-alone applications (*ScopiraPA*, *EvIdent®*, *Opus*)

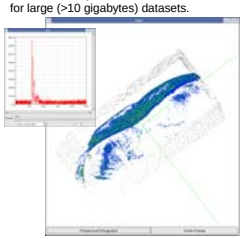
EvIdent®

Project based application for the detection of "novel" activations (time courses) in functional magnetic resonance neuroimages.



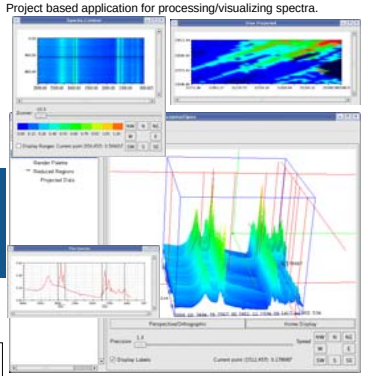
BigVol

A 64-bit and OpenGL-based 3D visualizer for large (>10 gigabytes) datasets.



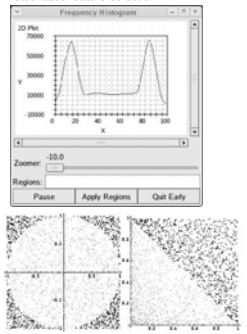
Opus

Project based application for processing/visualizing spectra.



SFS

Parallelized classification methodology for stochastic feature selection.



Graphics Subsystems

- ***coreui***: basic **GTK+**-based widgets, window classes, and layout managers.
- ***uikit***: builds on coreui to provide more complex visualization widgets and views (eg, plotters, image viewers, scalable matrix editor).
- ***lab***: provides API for rapid development of algorithms that need graphical output. Allows main thread to do the computations while a background thread handles GUI event loop.
- ***coregl***: builds on *coreui* to provide foundation for building 3D visualization widgets. Uses **GTKGLExt**, a small library that allows GTK+ apps to use the industry standard **OpenGL** API.

RDP

Relative Distance Plane: fast projection technique for high-dimensional data available in ***ScopiraPA.***