The Trybos Project

15-May-1997
Rob Kennedy

A) Goals and Platforms
B) Design and Concepts
C) Progress to Date
D) Short-Term Priorities
E) Future Plans....

Trybos Goals and Platforms

- Object-Oriented C++ access to existing data - improve maintainability, CPU efficiency, data model flexibility
- Read/write files which YBOS can read/write - can read old data and mix Trybos/YBOS use during transition
- Permits low-overhead named data accessors - no more hard-coded offsets or parameters into IW array required

- Geared towards Standard C++, Unix/POSIX platforms - verified to operate with SGI IRIX, Intel-based Linux, DEC Unix, and built on other systems - not too Unix specific
- Most development done under Linux with GNU C++ 2.7.2
- Can read/write "VMS" and "Unix" byte order, can write "Unix" byte order
- Oriented towards sequential access, limited YBOS format

Design and Concepts

- TRY_Abstract_File = a collection of TRY_Abstract_Records
  Different implementations for different media, formats, etc.
- TRY_Abstract_Record = a collection of TRY_Generic_Banks
  Different implementations for different I/O/search trade-offs
- Framework/AC++, user is passed a pointer to an abstract record
- TRY_Record_Iter_Any/Same = locates a bank within a record,
  Any goes to next bank of any name
  Same goes to next bank of same name
- TRY_Generic_Bank = a contiguous block of data with a header, type description, and user data sections
- Specific bank classes describe named banks, ie. CMUO_Bank, and inherit data accessors from base class TRY_Generic_Bank
- TRY_Bank_Key = (TRY_Bank_Name, TRY_Bank_Number)

Progress to Date

- Overhaul of record iteration (Trybos v3.2) - changes to interface simplified their use and improved CPU efficiency
- CPU Trybos test/CPU YBOS compare = < 50% (was 70%)
- Overhaul of generic bank and specific bank tools (Trybos v3.3) nearly completed, patches to be released in next week
- Data driven specific class generator program released
  139 specific bank classes released in new "Banks" product.
  These classes have generic data accessors since this program can only extract the type of user data elements, but not the purpose or common name. Mono-type and fixed-size simple mixed-type banks treated. List on WWW.
- Adaptations to SoftRelTools & CVS, FNAL C++ style, Framework/AC++ I/O and F77/YBOS co-existence, ...
- WWW page, User's Guide v1.0 (CDF 4174, latex/html)
- **http://www-cdf.fnal.gov/upgrades/computing/projects/trybos/trybos.html**

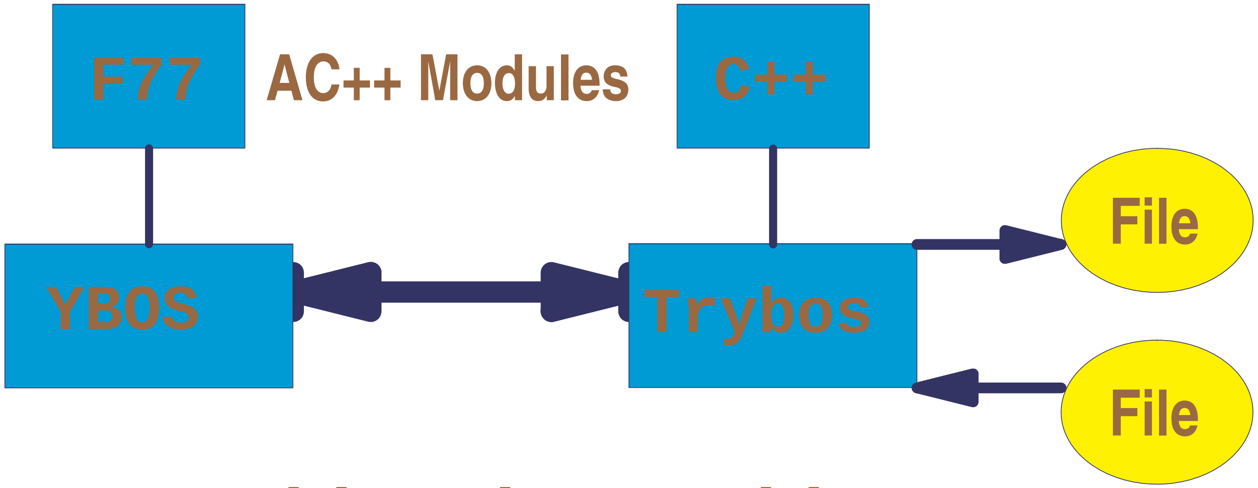
Short Term Priorities

- Completion and integration of banks overhaul
- Specific bank class support - delegate task of defining appropriate names of data elements in existing bank classes
- Specific bank class support - help in development of bank classes for banks with more complicated structure, prioritized by input from physics groups
- Specific bank class support - develop base classes to describe banks with common structure - LRS 1879 reformatted TDC 'D' banks like CTCD, CMPD, CMXD, ...
- Documentation - complete User's Guide, begin expert's Reference Document (overhaul of CDF 3518), latex/html
- Training - Intermediate C++ class using Trybos and an AC++ module as example - one week in July/August 1997
- Training - Tutorial - creating/refining specific bank classes

The Trybos Project
15-May-1997
Rob Kennedy

Future Plans: YBOS Emulation

Current State of
YBOS -> Trybos
Emulation
Transition



- We can now mix F77 modules and C++ modules...
- All I/O operations are performed by Trybos
- Framework/AC++ passes data between Trybos and YBOS internal data structures
- Next, define subset of YBOS to support in the emulation: Do not need I/O, Aliases, YBOS array utilities, for example

YBOS Emulation (cont.)

- Fortran-77 modules limit implementation: They see one array "IW" and contain hard-wired offsets into this
- Most operations very simple to emulate: BLOCAT(), BCOPY(), BTYMAK(), BNEXT(), BDATA(), BVALID()
- Need to develop and integrate concept of a bank name set, related operations and relationship to I/O --- also simple
- Most likely to continue to link banks with bank numbers, prefer both bank name and number to be stored as a key
- Choice: support or remove work bank use in F77 code --- work banks = dynamic allocation, better alternatives in C++
- Choice: maintain data in single event array for both C++ and Fortran (current choice) ---OR--- C++ interface to banks through intermediate data structure (offsets are no longer used) and searched for banks are copied back and forth to a Fortran work array for Fortran module use.

Other Future Plans

- Improve error detection, reaction, and logging
- F77 and C++ bank description header files - synchronization, maintenance, consistency,....
- Bank class use design by higher-level analysis objects - analysis objects use banks object for persistence, and are not specializations of banks to obtain persistence attribute
- More file and record implementations - TRY_SHM_File, TRY_Tape_File, TRY_CDFDB_File, TRY_Broken_Record (holes in single event array Level3), TRY_Listed_Record (multiple event arrays, optimal for C++, but is it still F77-compatible?)
- Explore concept of a TRY_Abstract_Dataset - use with lightweight event "databases" and other storage models
- Continue exploring issues related to ODMG-93, a Trybos-OODBMS interface, and continued F77 support