Answers to Questions (Hall A)

Q: *What number times to reprocess was assumed in the storage and CPU estimates?*

A: In general, 3 times. See the spreadsheet at
`https://userweb.jlab.org/~ole/HallA_12GeV_SciComp_Resources.xlsx` go

Q: *Discuss your tools for software quality assurance*

A: Methods & tools

- Design rules
  - Work within C++ analyzer framework (use prescribed functions to do things)
  - Go through standard APIs for common tasks (e.g. database access)
- Consistent coding standards
  - In general, we try to follow the ROOT coding conventions closely, awkward as they sometimes may be
  - variable and function naming
  - indentation and bracketing
  - commenting requirements

Answers to Questions (2)

- Revision control system (CVS)
  - restricted commit permissions
  - automatic email notifications to experts/reviewers
- Code reviews
  - Commits checked by experts at commit time (upon email notification)
  - Branch merge review before merge
- Testing
  - Critical inspection of online replay results
  - 12C optics runs that almost all experiments do
- Release Management
  - Only expert(s) are allowed to make official releases
  - All changes/contributions must have been reviewed line-by-line
  - Must compile cleanly (without warnings!) on all supported platforms
  - Must compile cleanly against a set of relevant ROOT versions (viz. those that experiments in prior 3 years have been using in in the counting house and on CUE)
  - Must not introduce new minimum software version requirements or new library requirements unless approved
  - Central binary installation in counting house and on central systems to reduce number of homebrew compilations (at least on site)

Answers to Questions (3)

Ideas for improved software quality assurance

- Define set of reference data, replay setup, and reference results
  - Try to reproduce after any significant software change
  - Allow tolerance for inevitable rounding errors
- Regular (annual?) code reviews using standard tools such as source code standards compliance checker (lint)
- As needed: performance checks with standard tools such as `valgrind` and profiler