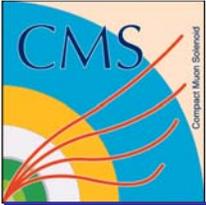


The Charge



- From Jim's email with instructions for the review:
- "The time limit for this review greatly restricts the scope. Harry and I discussed looking at a few aspects:
 - ◆ bigger structural problems (has the problem been decomposed well? is the design reasonable?)
 - ◆ looking for bad memory use
 - ◆ areas of code that might cause performance problems
 - ◆ areas of code that might have maintenance or stability problems
- With regards to the last two items on the above list, Harry and Kurt will point out specific areas that could be problematic. Harry also mentioned that one useful type of comment would be to call out areas that need more detailed review."
- The following slides list some areas that are candidates for your consideration.



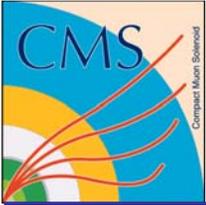


Possible Areas for Review



- Handling of exceptions in threads and subsequent communication of problems to other parts of the application and to operators.
- Minimizing the performance impact of web-based monitoring while still providing up-to-date statistics.
- Ways to checkpoint or monitor the various threads and resources.
- Ways to enable debugging operations and/or output on-the-fly with minimal performance impact.
- Modifications to the management of the threads to avoid duplicate methods to fetch data or exposure of object internals.
- Places in the code where A) objects are copied, B) references are used, or C) shared pointers are used, and a different choice would lead to better performance or better memory usage or more reliable operation.



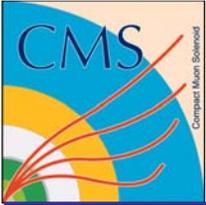


Fault tolerance – existing issue list



- Please see the tables in section 4 of <http://home.fnal.gov/~cheung/smreview2/smreviewdoc2.pdf>





Fault tolerance in fragment collection



- Some of the changes suggested in the fault tolerance list have recently been added to the fragment collection (FragmentCollector.cc). This includes checking for duplicate fragments, fragment numbers out of range, and missing fragments. We also now handle fragments that arrive out of order.
- With these changes, the handling of fragments and subsequent release of I2O buffers should be much more reliable.
- However, a similar issue exists in StorageManager.cc with regard to sending the “discard” message to the resource broker. In normal running, this message tells the RB that the SM has received all of the fragments for an event, and the RB can free up the buffer for that event.
- Since the receipt of fragments in StorageManager.cc is de-coupled from the processing of fragments in FragmentCollector.cc, it seems like we should avoid coupling between the release of the I2O buffers and the sending of the discard message. As such, it seems like we need to duplicate the fragment tracking code in StorageManager.cc
- Of course, a more reasonable idea is to create a templated class that contains the functionality to track fragments and “clean up” when full events have been received (or it gives up waiting for an incomplete event). This class could be used both in FragmentCollector.cc and StorageManager.cc.
- Thoughts?

