HLT Histogramming Service
Histmon Status

Ricardo Abreu[1] Tomasz Bold[2] Andre dos Anjos[3]

[1]CERN [2]UCI [3]UW

MWG Meeting
September 3, 2009

Summary of changes

Main changes:

- Maximize what can be changed with patches
- Disallow interactive binning commands
- **Parallelize histogram publishing**

Maximaze what can be changed in patches

- We want to be able to do most changes without touching public header files
- Limit public header files' contents to interfaces
- Implementation is outside of the public folder

How

The classes in the public header files are "shallow".

- They have only one private member - their implementation
- Objects of the "public classes" forward every operation to the implementation object that they hold.
- Every result is also forwarded back from the implementation to the caller.

Disallow interactive binning commands

Interactive Commands - commands received during run

Initial Commands - commands received before SOR

Binning Commands - those that change the binning of histograms (e. g. WidenRange)

- Histogram's contents are only filled after SOR
- Histogram's publishing only starts at SOR
- The methods behind the binning commands are not thread-safe
- Initial commands are processed at SOR - Safe!
  - No one else touching histograms

Parallelize histogram publishing
The Problem

- There are different types of histograms with different required publishing frequencies
- In `tdaq-02-00-02` frequencies weren't always being respected

because...

the established period between two consecutive publications could be exceeded while other histograms were being published

e. g.

- 2 types of histograms
  - Type standard - period 30s
  - Type express - period 5s
- In `tdaq-02-00-02` histmon did everything with a single thread
- If publishing standard histograms takes too long, express histograms aren't published in time

Parallelize histogram publishing
Quickfix

- Early solution applied as a patch to tdaq-02-00-02
  - No chance for big changes
- Before being published, a histogram is put in an ordered send queue
  - Ordered according to priority
  - Histograms have to be inserted at the right position
- Histograms are poped out of the queue to be published
- These two operations are interleaved
- **Low efficiency!**

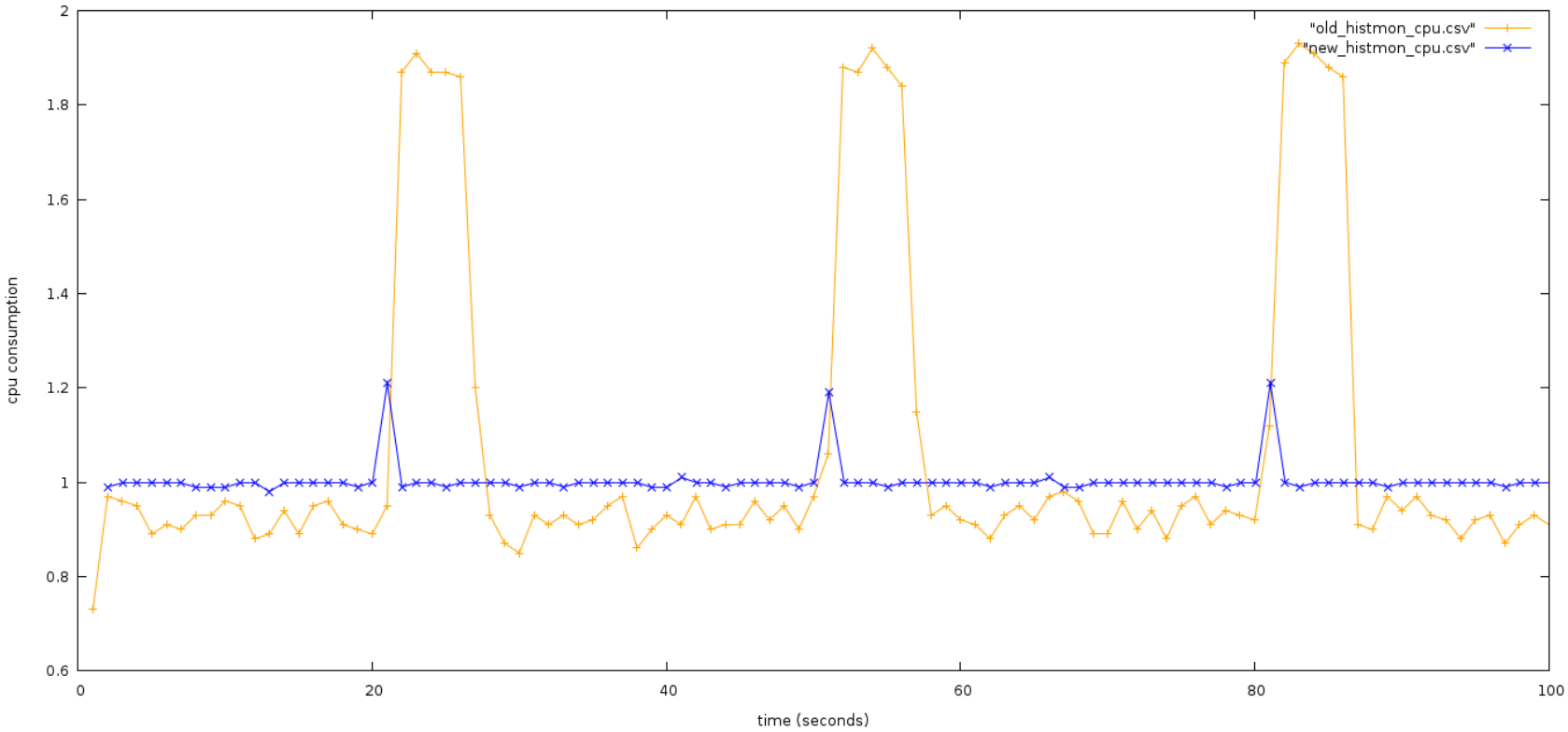
Parallelize histogram publishing
Final Solution

- **One thread for each histogram type**
  - Each of these threads roughly follows the control state machine
  - Each of these threads takes care of the histograms of its type
- One global thread for routing operations to other threads

We don't want too many threads per application!

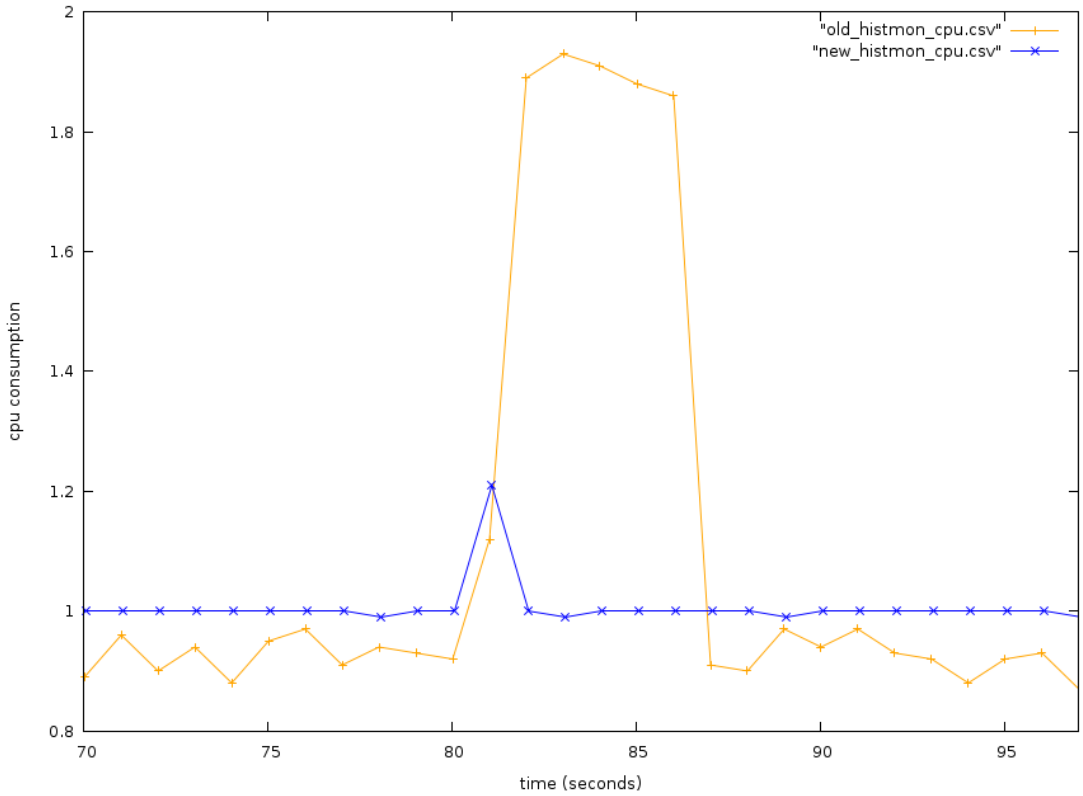
- Previous histogram types with the same publishing period were joined into a single type
- Some histogram types are meant only for some applications
  - **HistogramTypeUpdate class in the partition configuration has a new field that specifies the target applications**
  - Partitions and segments generated with PartitionMaker already take this change into account
  - **The default value is an empty list - no histograms in old partitions**

Obtained with athenaMT



- Peaks due to histogram publishing
- Fluctuations due to external factors

Results



- The Oy axis is the cpu consumption - $\frac{NumberOfClockTicksUsed}{NumberOfClockTicksElapsed}$
- Measured by intervals of 1s
- What matters is the amount of cpu used - **Area**

Conclusion

- More flexibility for fixing problems with patches
- Interactive binning commands are no longer allowed
  - no risk of undefined behaviour
- Histograms with different publishing periods are addressed in parallel
  - Publishing periods are respected
  - Better performance