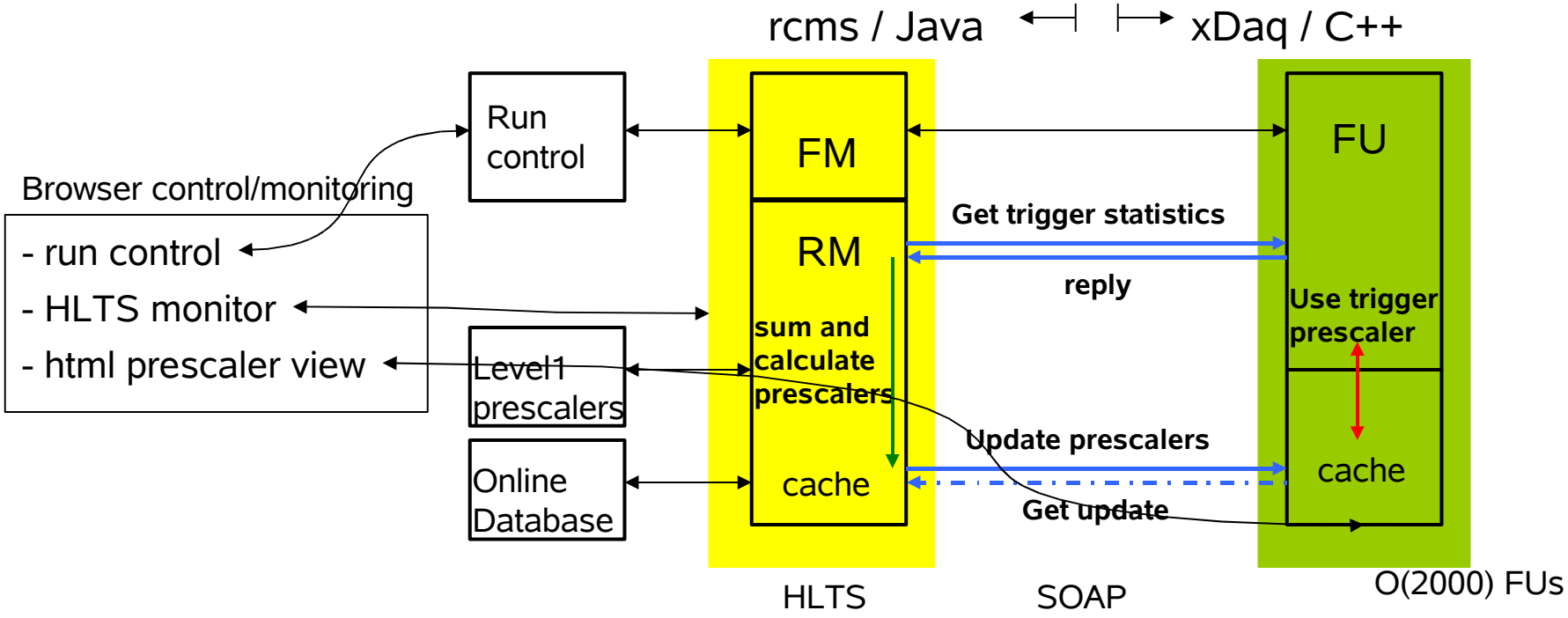
HLT supervisor status and plans

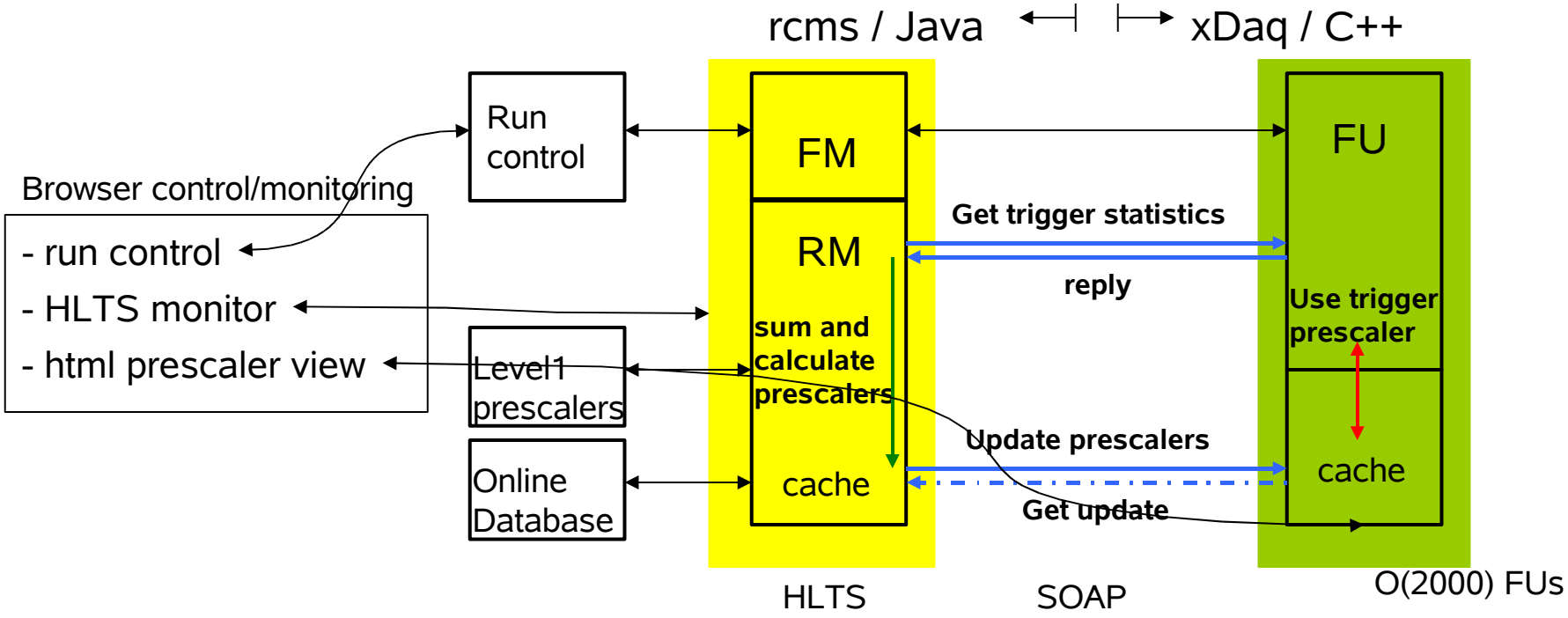
- HLTS requirements
  - Read trigger table from DB and store HLT tag to condDB
  - Distribute trigger table to FU before start of run
  - Collect HLT trigger rates and calculate/monitor prescalers
  - Control dynamic parameters such as prescalers
  - Receive L1 supervisor trigger statistics
- Currently concentrating on getting prescaler handling working
- Contents of this talk
  - Review status of prescaler work
  - Plans for the HLTS

HLTS prescaler status: design 1



- SOAP messages transport information (Get Update not yet implemented)
- FM - Function Manager implements run control finite state machine
- RM - Rate Monitor; polls trigger statistics, calculates prescalers, updates its cache, and pushes updates to FUs
  - If FM/RM has to be split for scaling a distributed caching pattern is needed, but this should not be problematic for LS boudary updates.

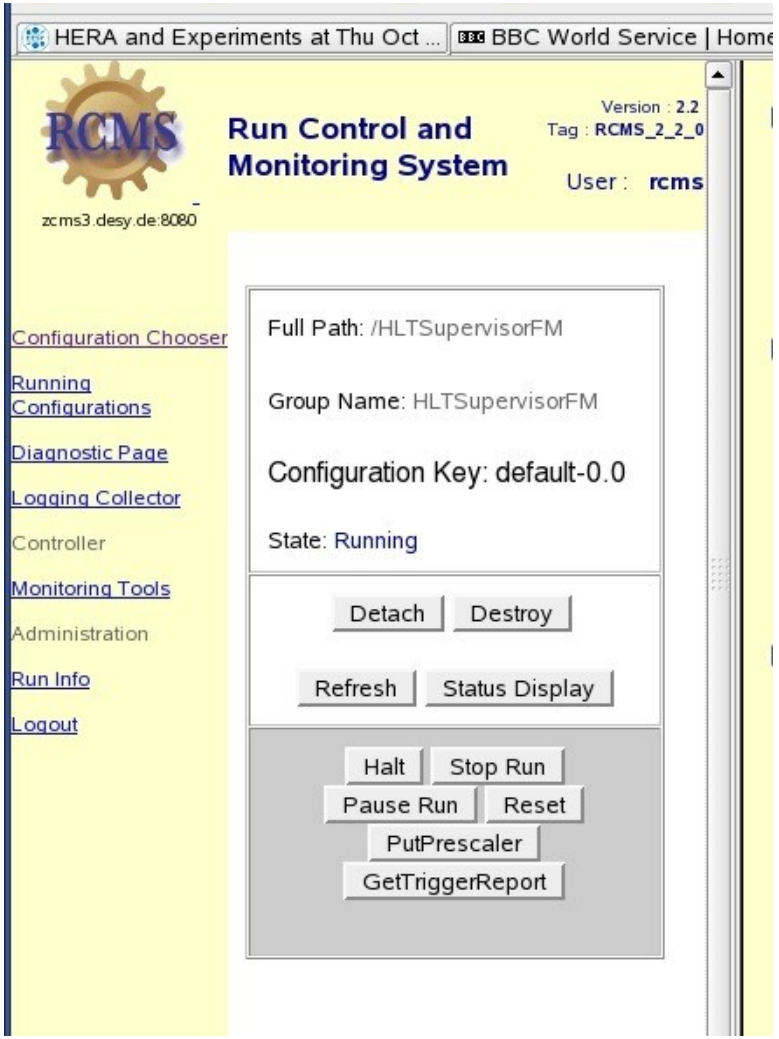
HLTS prescaler status: design 2



- Filter software accesses prescaler cache through service class (PrescaleService).
- Prescalers fixed for period of luminosity sections, typically 10 minutes.
- Prescalers used written to condDB
- Information exchange between L1 and HLTS to coordinate changes

HLT prescaler status: test implementation

- Test system
  - Currently used CMS software releases
    - Run control rcms 2-2-0
    - CMSSW-0-9-0 framework
    - xDaq 3.5 software
  - System consists of
    - run control
    - job control
    - HLTS function manager, including
      - Buttons for generating Update and getTriggerReport requests
      - Rate Manager which periodically polls trigger statistics, calculates prescalers (unity), and updates FU caches
    - FUEventProcessor (self generated data/trigger flow.)
    - Running on dual CPU CERN SL3 at DESY

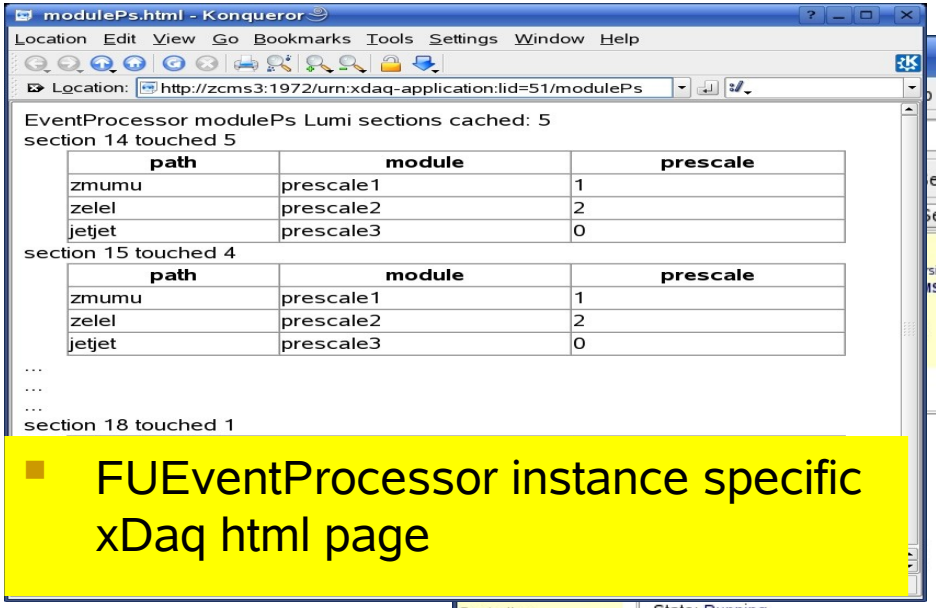


HLT prescaler status: PrescalerService class

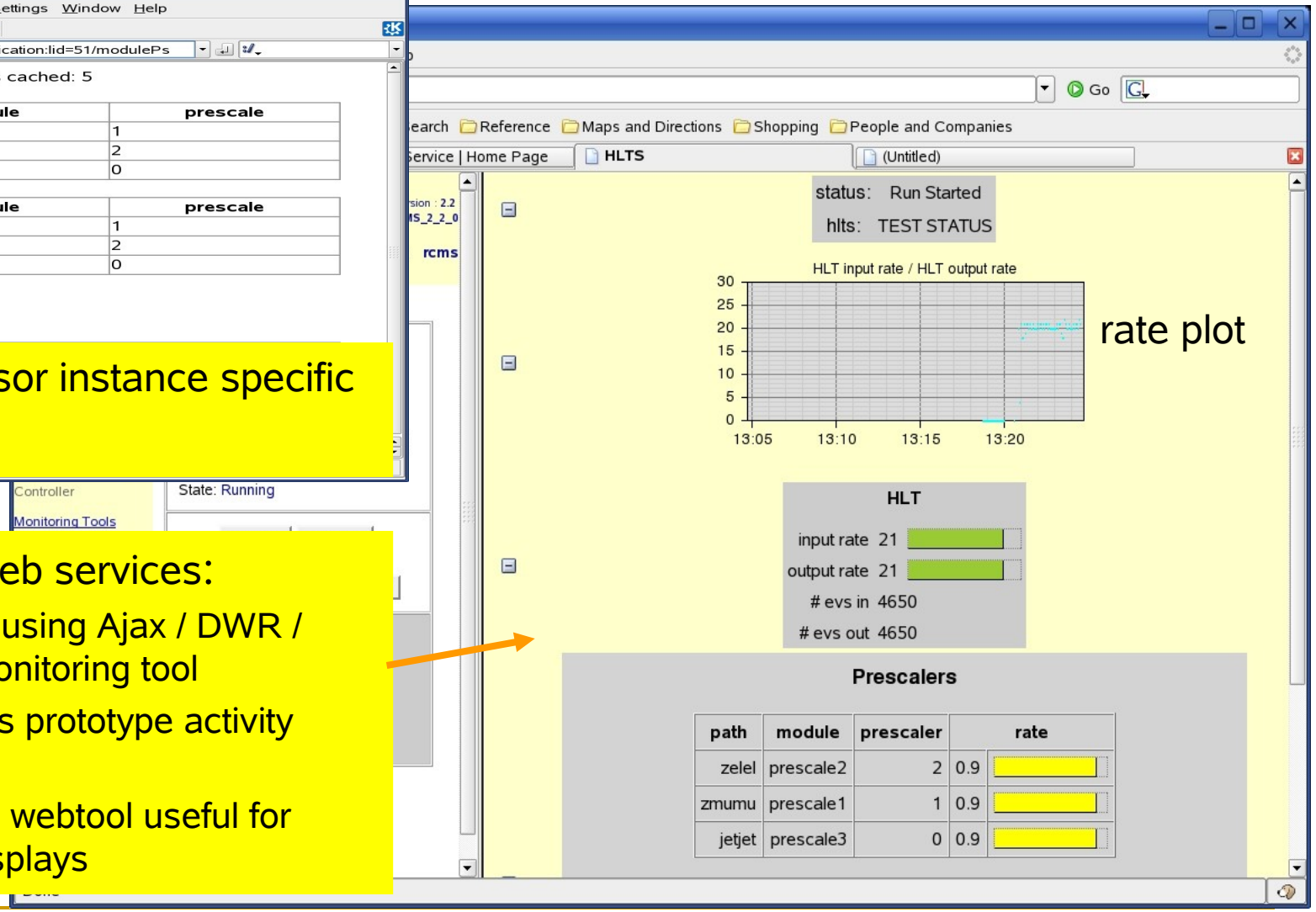
```
...
#include "boost/thread/mutex.hpp"

...

    class PrescaleService
    {
    private:
      Cache store;

    public:
      PrescaleService(const ParameterSet&,ActivityRegistry&);
      ~PrescaleService();

      ...

      int getPrescale(unsigned int ls, string path, string module);
      int getPrescale(unsigned int ls, string module);

      int putPrescale(string s);

      int sizePrescale();
      void listPrescale(xgi::Input  *in, xgi::Output *out);

    };
```

- methods
  - putPrescale
    - inserts prescaler information into cache.
  - getPrescale
    - Retrieves prescaler for given LS#, path and module name.
- Data
  - store contains last N LS# prescalers
  - Access to data synchronized using scope mutex lock
- Class usage
  - loaded in FUEP for filling
  - Used in Framework HLTPrescaler class to access cache
  - Wait for response from Martin.G

HLT prescaler status: monitoring



- FUEventProcessor instance specific xDaq html page



- HLTS Tomcat web services:
  - Javascript using Ajax / DWR / Web2.0 monitoring tool
  - Tool makes prototype activity visible
  - Underlying webtool useful for monitor displays

HLT prescaler status

- Has been implemented
  - SOAP messages including en- and de-coding
  - Filter Unit PrescalerService class
  - Rate Monitor caching of prescalers (unity calculation of new values)
  - Simple viewing html page for FU cache debugging
  - HLTS tomcat web services monitor using Ajax /DWR / Web2.0
- Test system functionally complete
  - Get Update message needs implementing
  - DB connection required
  - L1 exchange messages required

Plans for the HLTS

- Plans
  - Test system functionalty (almost) complete and works
    - Needs integration with HLT development at CERN
    - Validation of scaling to full size HLT
  - Next steps
    - Interface to online DB
      - RM is written in Java Tstore not recommended
      - RCMS use JDBC
      - Use a JDBC based solution, must be validated with DB group
    - Interface Level 1 information
    - Other HLTS requirements
  - The above were addressed during the HLT workshop 31.Oct.2006 at CERN