Flight Software

How we spent our summer vacation

• SOFTWARE
  - Code Management, CMT/CMX
  - Low Level Utilities
  - L2 Filtering Studies

• HARDWARE
  - Architecture & Bandwidth
  - Rate Counters
  - Trigger
Code Management - CMT

- Decided to go the same route as GSW
- After a long meeting, Autoconfig/Autotomake was chosen
  - After some time, this decision was changed to CMT
- Added to & modified CMT quite a bit
  - Still a young product, i.e. it has some bugs
  - Its concept of shareable images is crude
  - Mainly addresses the build process, not the life cycle
  - Does not understand a cross-develop environment
- Three prong attack
  - Fix CMT bugs by petitioning the author
  - Fix CMT bugs ourselves
    - Have maintained a minimalist approach
  - Augment CMT using the supported interfaces
    - Added real shareable & cross-develop environment support
Code Management—CMX

- Adds a layer on top of CMT to address life cycle issues
- Defines a model to
  - Begin a new package and test it privately
  - Expose the package to other (ab)users, a development stage
  - Support it as a finished product, a production stage
- Also adds support for 'mixing and matching' package versions
- A fine user's manual (courtesy of Tony Wait) is complete
- What it does not do
  - Unfortunately, it does not support NT
  - This is a real problem
    - We've promised that we would export software from the FSW group to GSW
    - NT is very different from Unix and we are a small group
    - Too much time has been devoted to this already
Low Level Utilities - Pieties

- Personally, I don’t like VxWorks
  - Tries to be all things to all people
  - Sometimes dangerous

- Would like to use a few well-established inter-thread methods
  - Typically a group of threads are bundled to form a pipeline
  - These provide the elastic as data moves through the pipeline
  - They remove a lot of the generality of the VxWorks stuff
    - Efficient support of single writer/single reader common in data pipelines
    - Can take advantage that upstream threads are generally higher priority than downstream threads
Low Level Utilities

- These fall into two categories
  - Interlocked memory managers
  - Inter-thread communication/synchronization tools

Memory managers
- Limit to lightweight, thread-safe, interlocked managers
  + Some resource wait support (read flow control)

Inter-thread communications
- Supports simple mailbox and queue mechanisms

Inter-CPU stuff has not been addressed, mainly because we don’t know our physical layer
- If it’s Ethernet,
  - May be TCP/IP
    - VxWorks TCP/IP implementation is not that good
  - Perhaps raw Ethernet
L2 Filtering Studies

- Looked into the CPU cycles expended
  - Goal was to estimate the number of CPUs required
  - Not to produce the final algorithm
- NRL explored Hough Transforms
  - Elegant, but costly
  - 800 usec/event on a 100MHz 603
    - Consumes the entire processing budget, based on power constraints
- SLAC explored
  - Simple road finders
  - Bit-based methods
  - Template-based methods
L2 Filtering Studies, continued

- L2 goal is to eliminate obvious cosmics as quickly as possible by finding a 'track' and stabbing an ACD tile.
- Given the low multiplicity and low noise:
  - Simple road finder seems adequate.
- Basic 3 in-a-row finder is very fast:
  - But permutations could be a problem.
- Tried on CHIME beam test data, both γ's and p's:
  - Permutations are not a problem in this data.
- Timing is roughly <1μsec to find a three-in-a-row.
- This is not the whole story:
  - To give one greater confidence, the track should be extended through more layers (guessing, ~ 5x this time).
  - Haven't stabbed an ACD tile (should be fast on this scale).
  - Does not include the unpacking time (not fast on this scale).
L2 Filtering Studies, continued

- The unpacking has proved to be the limiting factor
- This takes 10-20\(\mu\)secs, thus representing the bulk of the processing time (estimated for a 603@100MHz)
  - Algorithm is within 20-30% of just touching the memory
  - Only way to go faster is to reduce the 'touched' data
- Because of this, turned attention to improving the input data format to be more software friendly
  - Not a trivial task to maintain readout speed and stay within power and space constraints

- Bigger issues
  - What is the rejection rate of L2 filtering?
    - Seems generally agreed to be \(~5-6\)
  - What's next?
    - Still need another factor of \(~5-10\)
    - How long does this next level of filtering take?
Unpack Times

- Times are for a 604@ 300MHz, multiple by 3-5 for 603@ 100MHz
Architecture and Bandwidth

- The basic architecture has not changed
  - Essentially 4-8 CPUs being fed complete events
- The necessary bandwidth is still a topic of debate
  - Normal running design point is ~1KBytes @ 10KHz
    - Results in 80MBits/sec
    - Current architecture has 120Mbit/sec pipe per tower
    - Ideally this gives 320Mbits/sec
      - Practically will likely achieve only 50-75% of this
  - Is this enough to deal with GRBs?
    - Assuming 10K realγ’s (within 100m sec) arriving once a second for 5-10 seconds, then likely not.
    - More bandwidth or more buffering? Where?
    - Both add complexity and, therefore, risk to the design
The CPU likely will also need increased buffering
  - Cannot deal with this rate of γ’s in real time.
  - At 1K byte/event, 50K events consume 50M bytes
    • Of course this gets spread over N (≈4-8) cpus.

Question?
  - How much do we allow the GRBs to drive the design?
Rate Counters

- Here we need some help
- The DRD calls for
  - Reading these at no more than 16 Hz
  - An accurate time stamp when these counters are read
- Issues & Questions
  - The data volume is non-trivial
  - Without assistance from the trigger, maintaining synchronization across the instrument is difficult
  - Can the rate counters contribute to the dead time?
  - Should the rate counters themselves be dead time less?
    - Implies double-buffering, which implies complexity
  - Just what is the science/diagnostics that these address?
  - How much complexity are we willing to absorb in accommodating these counters?
This is another trouble spot

Current trigger design is simple and robust

But has limited capacity to implement global features
  - e.g. Tower vetos

Once one goes to a central trigger system
  - The complexity goes way up
  - The flexibility goes up
  - Is this necessary in order to
    - Accommodate present needs?
    - Absorb future gotcha’s due to changing demands?
Manpower

- Essentially 3 full time people available to write the code
- With 2 part time people to help on interface issues
- Geographic distribution leads to less than optimal efficiency
- Skill and experience mix is not ideal
- Overhead (paperwork) and budget constraints, well you know the sign
- Management lines are not crisp

Examples

1. < 2 weeks to produce unpacking and track finding results
   - This was all I could muster from March till now

2. CMT/X has absorbed all of Tony's time

3. NRL is unable to buy long lead time items for the balloon flight
Manpower, continued

- We are asking for trouble
  - We are already seeing signs of this
  - We cannot continue like this