Online Software Issues

- Which tasks will be needed for SVT
  - Operation
  - Monitor
  - Diagnosis
  - Debug
- How will they be implemented. By whom.
- How do we interface with run_control.
- How do we interface with online/offline DB, offline data analysis ...
- How much CDF “standard” online software is re-used
- What will run on VME CPU, what on local host(s), what on remote institutions
- Which local computing resources will we need (CPU, disks…)

Present Status

- Let’s be honest: we do not have a plan.
- Today’s goal:
  - List of tasks
  - Estimate of manpower needs
  - Solicit volunteers
  - Assign responsibilities
  - Agree on a baseline for basic interface issues with the rest of the world
  - Define a time frame for a plan
- This talk:
  - My best understanding at the time = lots of wrong ideas to stimulate you to come forward with the right ones (even if not immediately)

Tasks

- **Operation**
  - define constants to download
    - how often do they change
    - do they change during the run/store ?
  - Check beam position before data taking.
  - Do we re-compute patterns at this time ? Better be prepared to.
  - Monitor beam position during data taking
    - adjust dynamically impact parameter measurement ?
- **Monitor**
  - goal: make sure SVT is working
    - tentative spec: check every hour that it is OK to 1% level
    - **need** 10K events accepted by SVT **and** 10K rejected
  - is the CO tool

More Tasks

- **Diagnostic**
  - If monitor spots a problem, find out where it is in such a way to allow the data taking to resume correctly with “simple” actions:
    - download proper constants in place of erroneous ones
    - swap bad board
    - push loose cable
  - is the ACE tool
- **Debug**
  - For detailed problem finding/solving. Pinpoint problem source, identify possible hardware failure and/or real bug in the hardware or software of SVT
  - is the SVT expert tool

Run_Control interface

- Data to be downloaded must be ready in before run start, ideally SVT has been fully configured and downloaded after power up.
- Run_Control must have a way to check (force?) this
- Run_Control must start SVT monitor process(es)
- After some data tacking (1 min ?) Run_Control must query SVT for a check on beam position.
- If beam position has to be adjusted, data taking has to continue to give feedback. If SVT parameters needs to be changed run has to stop/restart with a different number (does it?).
- SVT severe errors are handled via CDF Error/Recovery/Start protocol
- No need for software message passing during run.
- At run end Run_Control signals SVT monitor process(es) to send statistics for proper storage (DB ? EndRun record ?)

Online vs. Offline

- Offline analysis will want to validate SVT trigger, compute efficiencies etc. Exact information of SVT configuration at the time data were taken must be available.
- There may be drawback. E.g. would prevent dynamical adjustment for beam movement by <d> subtraction to smooth out small oscillations on the minute time scale.
- SVT geometry must be made to relate with offline one. Not an understood problem yet. What if two beam monitor program report differently?

SVT monitoring

- Will likely need 2 monitor programs:
  - ➢ one standard consumer runs off the Level3 data to validate SVT triggers (make sure when we say impact parameter is high, it really is. Do not waste trigger bandwidth on fakes)
    - ☞ let's not worry about this one now
  - ➢ one special SVT Spy Buffer monitor program (SVT monitor) looks at events rejected by SVT to make sure they really had to get lost. SVT reduction is O(10^3), and these events do not make it past Level 2 in other ways… for each event we see in Level 3, there are 999 we will never see. Make sure efficiency is under control.
    - ☞ Spy Buffers keep history of all data processed by SVT
    - ☞ Spy Buffers are read from VME without interfering with trigger processing.
    - ☞ Spy Buffers are the favorite place for SVT monitor

Monitoring beyond SVT

- Making sure that the hardware is doing its job is not enough
  - ➢ make sure SVT is always well tuned to the external environment

- is the beam moving ?
- is the detector (SVX) moving ?
  - ☞ we do not expect it, but… better check, maybe only seldom
- must monitor SVX strips noise:
  - ➢ a few very noisy channels could jam SVT
  - ➢ maybe monitored elsewhere ?
  - ➢ need some way to feed it back to Hit Finder

SVT monitor vs. TRIGMON etc.

- SVT monitor does not run from CDF DAQ data stream
  - ➢ there are good reasons for this
    - ☞it is done this way and can not be changed
    - ☞event lost by SVT will not be in DAQ stream
    - ☞need to monitor L1 triggers that fail L2
- Will need our own little DAQ
- Still can try to make it look like standard DAQ
  - ➢ special AC++ input module
  - ➢ L1 data collected into events and analyzed by standard AC++ consumer
  - ➢ use standard CDF tools for histograms, message passing, remote connections
- Can stick to standard approach with (SVTMON?) code running off CDF DAQ stream.

Spy Buffer Monitoring. How good is it ?

- Time to read Spy Buffers:
  - ➢ 128Kwords/Buffer, 2 Buffer/Board (typ.), 100 Boards
    - ☞$\leq 3*10^4*10^3$ words ≈ 100 Mbytes ⇒ few sec's
  - ➢ There is lot of redundancy (two buffers per cable!), most tasks will not require reading everything.
  - ➢ Each Spy Buffer full dump has several events in it (>100)
- Run full SVTSIM:
  - ➢ few ev/sec on ALPHA 255MHZ, assume 10Hz on RunII CPU
- SVTSIM timing matched to Spy Buffer reading: will fully check 10^4 events each hour or better, just the 1% we asked for.
- Spy Buffer can do a lot more: error monitoring and reporting will heavily use them (separate session), still:
  - ➢ hope we are not swamped by errors
  - ➢ SVTSIM and error monitoring can go in parallel

More on Spy Buffer Monitoring

- Spy Buffers also provide tool for simple monitoring
  - ➢ collect statistics of occupancy, YMON style plots
  - ➢ check for non-severe errors and count/list them
- This will not require SVTSIM.
  - ➢ Accuracy will be limited by readout rate
  - ➢ will benefit from parallelisation of various crates and exploitation of local CPU
- SVTSIM is a complex piece of obscure C code, doubt we will want to run it on local CPU (but it would allow parallel execution)

- Spy Monitoring will require heavy usage of network and CPU. Better have our private ones.

Spy Buffer as Beam Finder ?

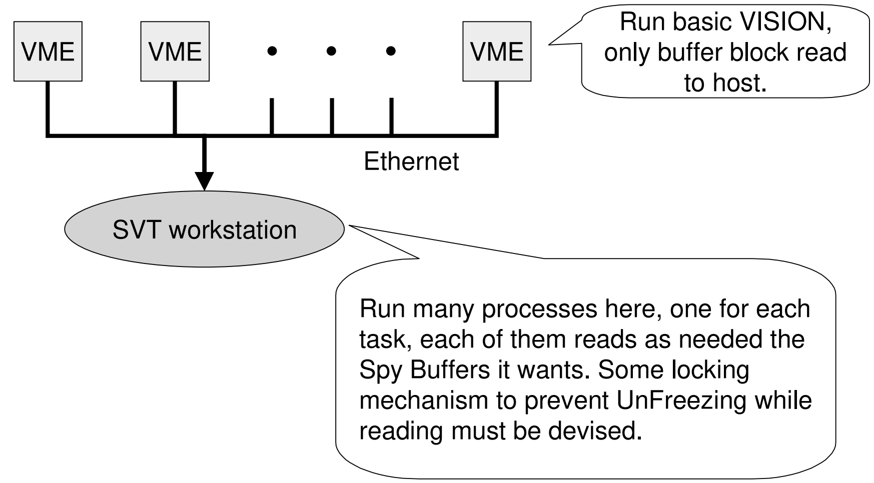
- This topic belongs to another workshop (**SVT & the Beam**)
- Just a place holder for more possible software needs
- Simple algorithm can track beam position parameters in real time and feed back into SVT constants (not fully proven), need ~10K tracks.
- All SVT output tracks go through the Spy Buffer of the last MRG
  - ➢ Data flow into Leve2: a few tracks each 20 μsec ⇒ O(100KHz)
  - ➢ Spy Buffer is filled at the rate it can be read by VME CPU
  - ➢ Simple program in crate controller could look at half the tracks produced by SVT (50% of time Spy Buffer is written, 50% is read)
    - ☞ **fit beam center with ~100K tracks each second !**
- Not clear yet how to turn this into $(x, y, \theta_x, \theta_y)$ in accelerator coord, but “it must be possible”.
- CPU & memory needs still not defined (should be small).
- Would this interfere with SVT monitoring ?

Spy Buffer monitoring. How to do it ?

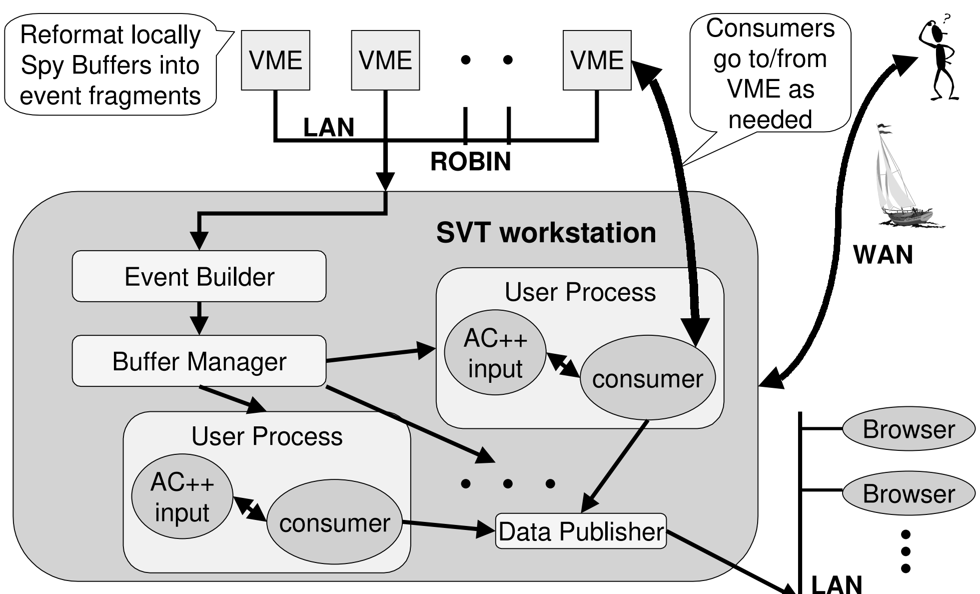
- Our own DAQ. Spy Buffers are read by VME CPU, asynchronously with data flow: SpyDAQ.
- Not a standard DAQ though:
  - ➢ One Spy Buffers holds many (up to O(1000)!) events.
  - ➢ Same event is in different places in different boards
  - ➢ part of current event is in FIFO's and not even read out
- How do we implement the needed functionalities ?
  - ➢ Remember also that
    - ☞same data can/should be used for several purposes
    - ☞different task may need (very) different amount of data
    - ☞this data are very useful for the expert, who is far away, across the ocean e.g. !
    - ☞Spy Buffer freezing can be "spontaneous" on error detection by the Spy Controls even without SpyDAQ requesting it

Simple minded SpyDAQ

- The **easy** way: do everything in high level software in the SVT workstation



Structured SpyDAQ (just a dream at present !)



Needed Hardware

- Disk.
  - ➢ Constants: See DataBase talk. E.g. 1MByte/pattern set. Need to have many sets ready (>100), e.g. precalculated for different beam positions. Best guess now: 2~3 GB.
  - ➢ Data: spool area where to keep Spy Buffer dumps for offline investigation: again a few GB.
- Computer.
  - ➢ One place where to gather pieces from all CPU's (event builder), and store data to be served to various monitor tasks (buffer manager).
  - ➢ Could be a piece of a large system
  - ➢ But maybe our software will take time to get stable and system will crash often… better a separate workstation. How powerful ? I wish I knew.

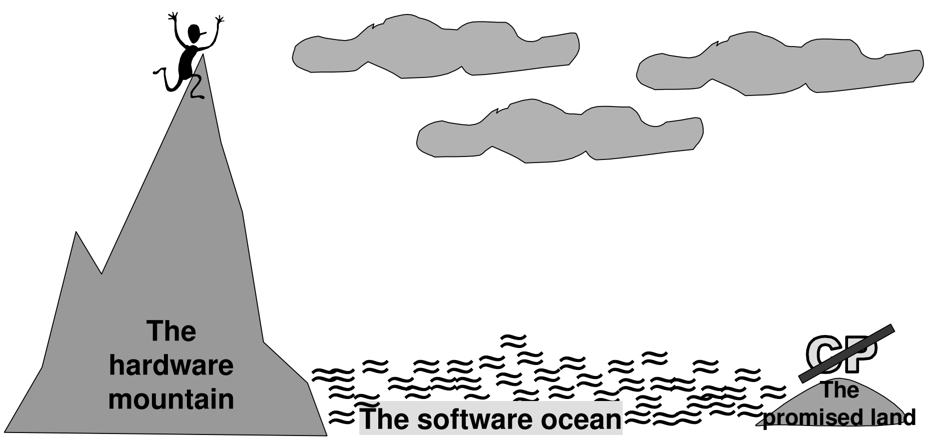
More Needed Hardware

- As long as monitor tasks will be standard AC++ consumers, they could (should?) run on same CPU's as rest of monitor tasks
- Some place where to run diagnostic programs for the expert… nothing special
- SVT test stand: our own crate next door with one SVT vertical slice
  - debug bad boards
  - can re-run SVX+XFT data from Spy Buffers (even "tape") through SVT hardware to understand (hopefully) even the subtlest problem
  - test code, patterns, programming etc. in the full system without interfering with data taking
  - provides hot swappable spares

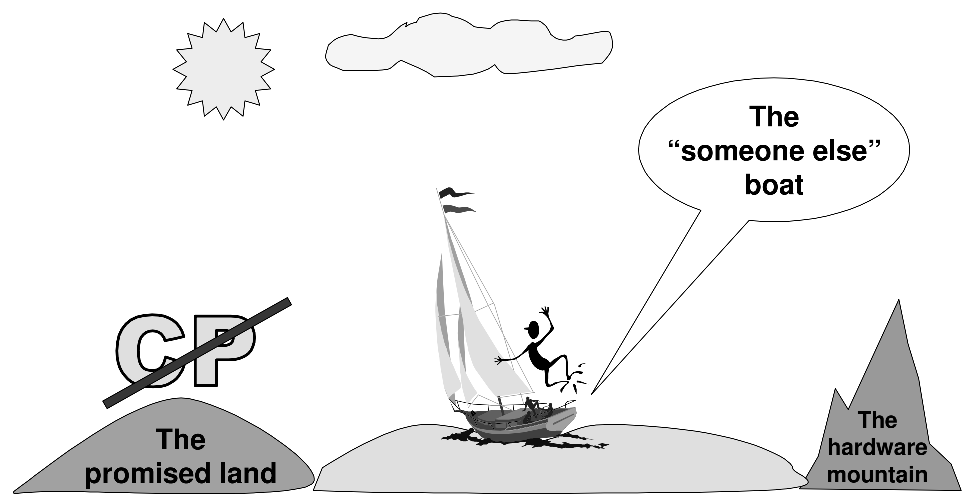
Needed Software

- Just all of it.
- This is to be filled (by someone else) after some work has been done
- for the time being
  - SVTSIM ~ there (need to be ported to AC++, e.g.)
  - Basic Spy Buffer handling within CDFVME framework (test, freeze, read, unfreeze) will be there by vertical slice time

Summary: the situation till yesterday



Perspectives: the situation tomorrow



Conclusions: what about the goals ?

- Today's goal:
  - List of tasks **: already too many ?**
  - Estimate of manpower needs **: what about 5 ?**
  - Solicit volunteers **: doing it right now**
  - Assign responsibilities **: better later ?**
  - Agree on a baseline for basic interface issues with the rest of the world **: too early ?**
  - Define a time frame for a plan **: winter's end ?**