

# EVB, Another Look

Stephen Adler

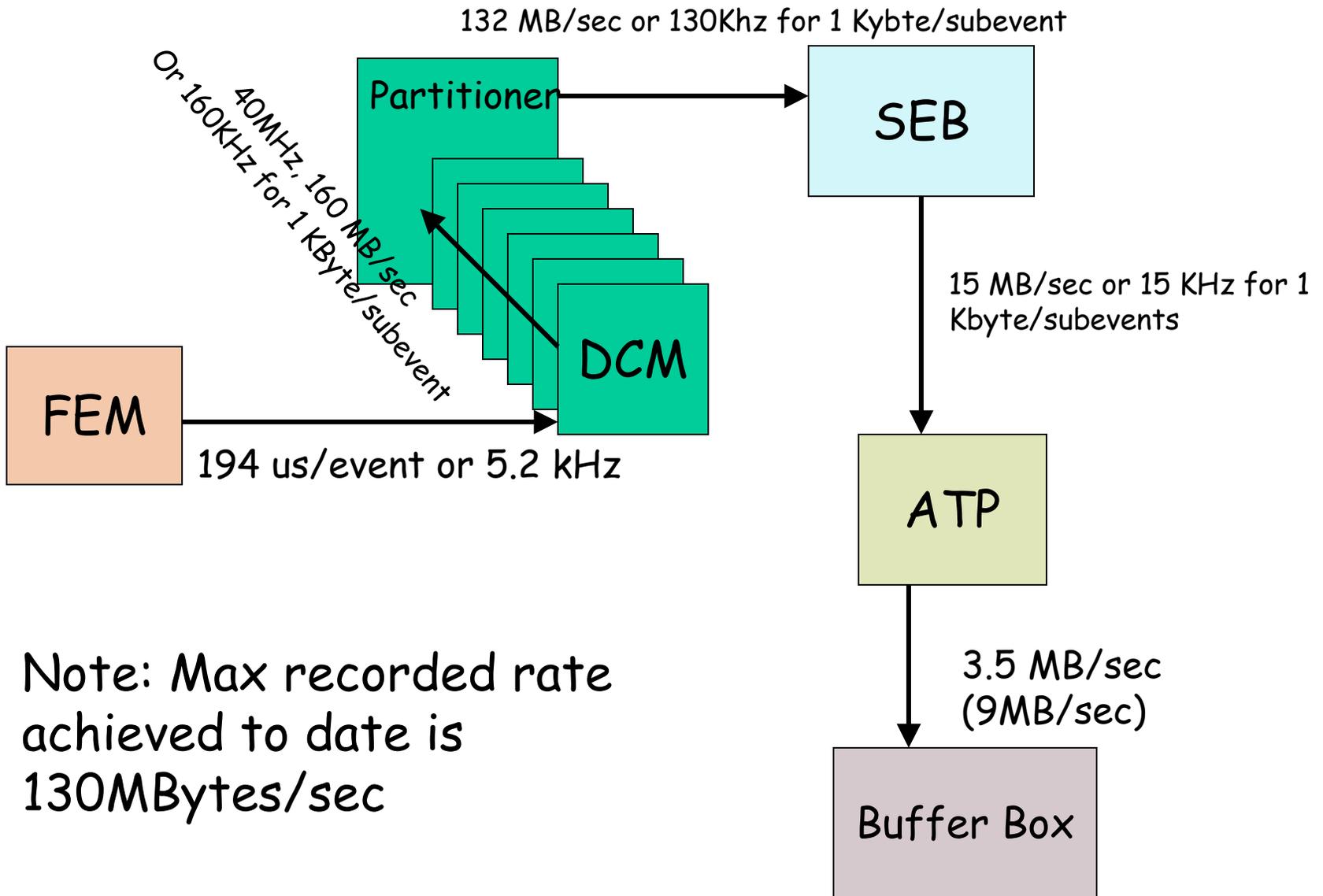
Post Run 3 DAQ Meeting

June 3rd, 2003

# Deconstructing the DAQ

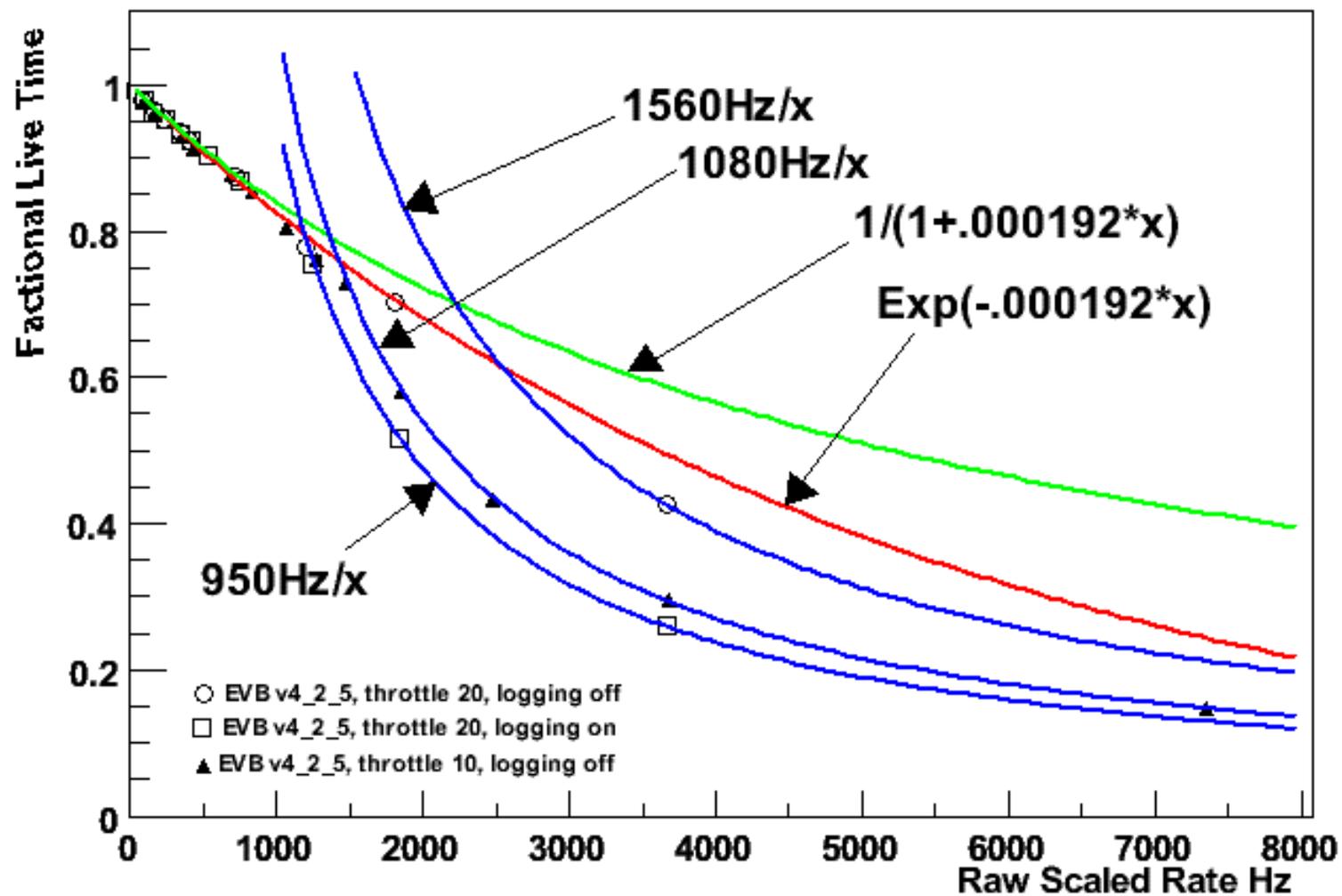
- Data Sampling in the FEMs
  - Runs at beam crossing rate of 9.4Mhz
- Data transmission from FEMs to DCMs
  - Limited to data packet length @ ~25ns/packet
- DCMs to SEB's via Partitioner
  - Nevis token passing bus
- EVB
  - ATM or Gigabit switch
- EVB to Bufferbox
  - Gigabit connections to buffer boxen

# Data Throughput



Note: Max recorded rate achieved to date is  $130 \text{ MBytes}/\text{sec}$

Live Time vs Raw L1 Rate



# Event Building

- 2 basic issues confronting the event builder ability to assemble events quickly. (I.e. @ high rate.)
  - Data rate through the “networking fabric”
    - I.e. what ever medium is used to connect all nodes in the event builder. (ATM, Gigabit, USB, Fiberchannel...)
  - Synchronization speed or latency of the data throughput
    - How fast can you get messages around the event builder nodes.
- Both data rate and message latency need to be balanced to achieve a high performance event builder.

# Rates

- The goal – 10KHz to the Level2 system
- Data flow
  - 100Kbytes/event means 1 Gbyte/sec to the Level 2 system.
- EVB Messaging must run some where between 10 to 100 us.
  - Most likely it will take more than one message per event, thus one needs to look for something closer to a 10us second solution.
  - Or implement message buffering, (I.e. one queue's up the data in the EVB components and then you process N events per message.)

# Event Building Communication Fabric Technology

- Available Technologies
  - 1000BaseT Network Switch
  - ATM
  - Mira Net
  - Reflective Memory
  - USB
  - Arcnet? (Lots of in house expertise)
- Messaging vs Data flow
  - Use the same communication fabric?
    - Pro: Less hardware, cheaper
    - Con: Message rate may be affected by data flow
  - Segregate data flow and messaging?
    - Pro: Potentially much higher event rate
    - Con: cost more, more hardware to install and configure

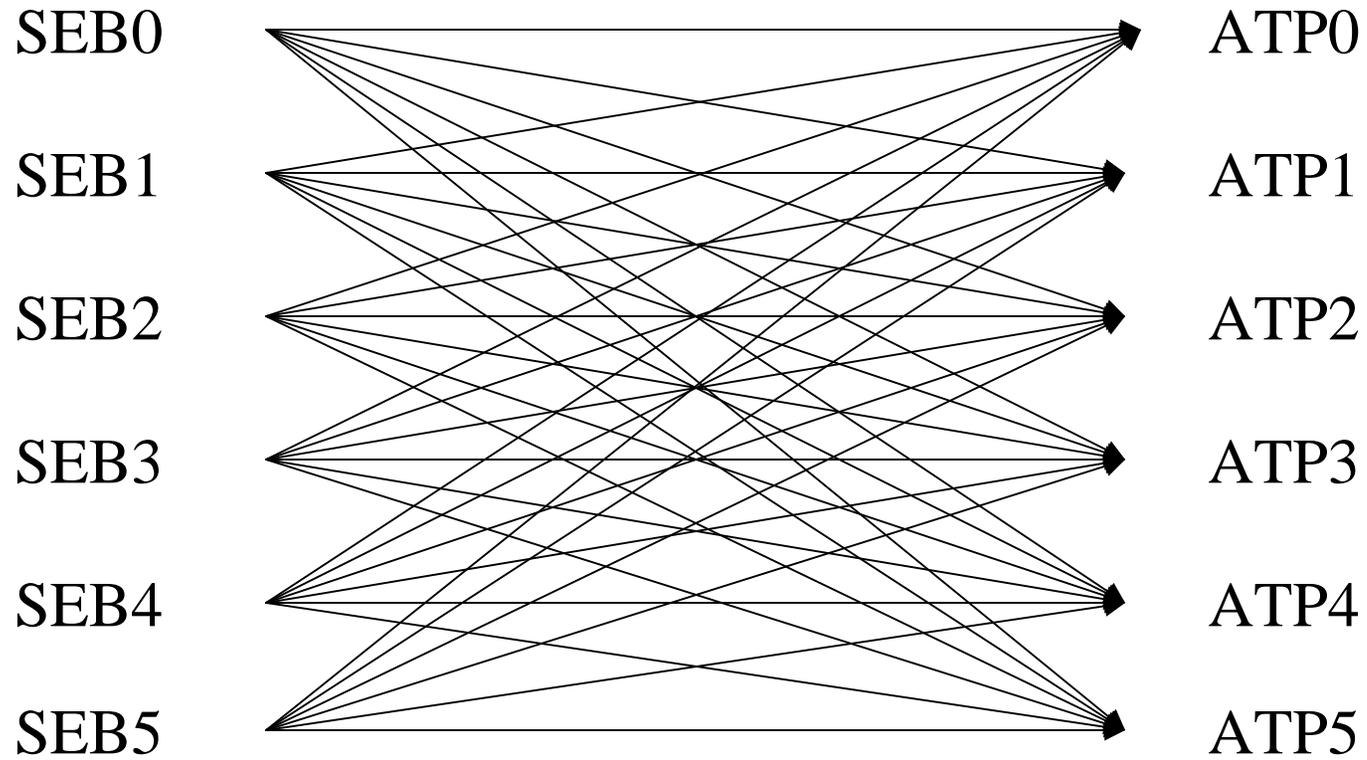
# Current Event Builder

- ATM based, Gigabit Upgrade.
  - Low level communication protocol used for ATM data transmission.
  - UDP IP communication protocol used for Gigabit data transmission.
- Data and message flow
  - ATM mode: both use ATM switch
  - Gigabit mode: Data over gigabit/messaging over ATM
- Heavy reliance on ATM flow control to enable the EVB to transfer sub events over switch.
  - When in ATM mode
- Use of multicast-like virtual circuits to reduce the amount of message passing needed.
- All nodes run Windows NT/2000
- Data integrity is built into the Event Builder design.
  - Data retransmission is designed into the hand shake between ATP's and SEB's.
  - Due to the use of UDP or low level ATM communication protocol.
- Current highest data rates achieved are ~2Khz
- Current highest data throughput achieved ~160Mbytes/sec

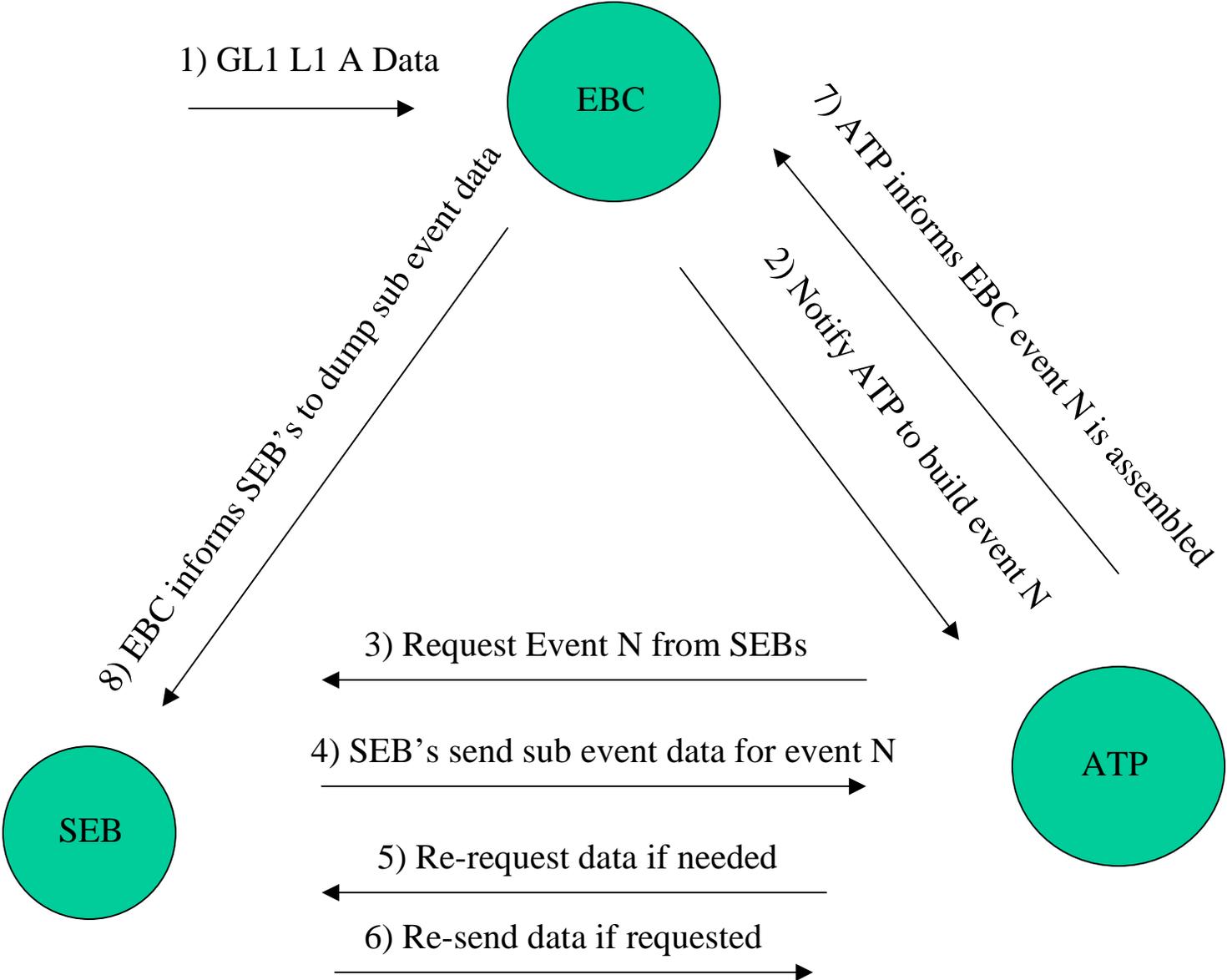
# Deconstructing the EVB

- GL1 SEB (SEB0) gets data for every trigger across all partitions
- For every L1A packet it gets, it assigns an event to an ATP, telling the ATP to get the data from a list of SEBs
- ATP goes off and sends a message to all SEB's requesting data for event X.
- SEB's send their data to ATP
- When done receiving all the data, ATP sends message to EBC telling it data for event X has been assembled.
- EBC tells SEB's they can dump their sub events for event X

# EVB Socket Topology



# Current Event Builder Design



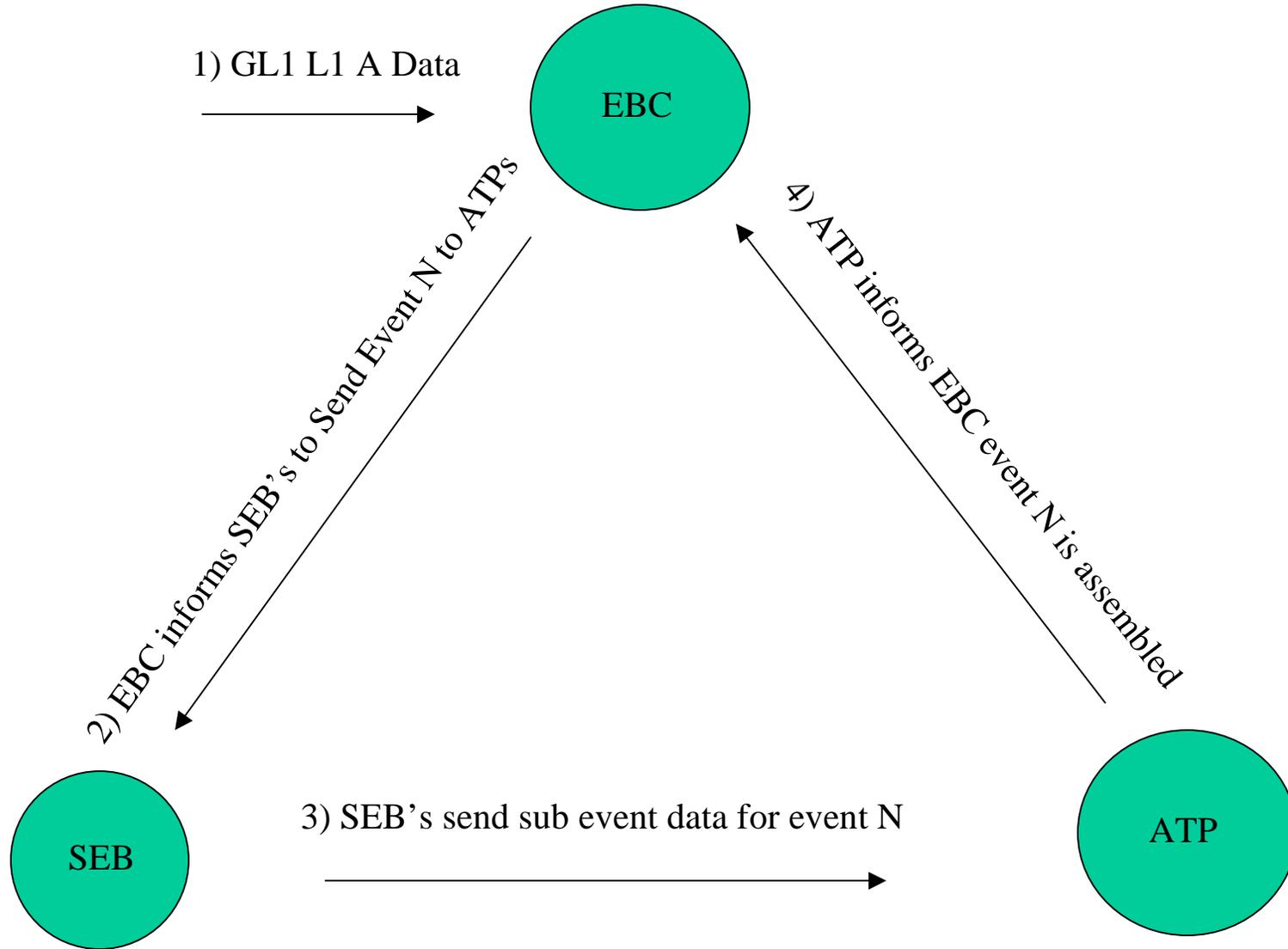
# Notes on the Current Event Builder

- Strong evidence that event rate is limited to message passing
  - Current upgrade to Gigabit will help
    - Free's up the ATM for message passing only
  - Message passing rate limitation is aggravated by the need to provide data re-transmission protocol.
- Strong reliance on tuning parameters to make the event builder work.
  - Data rate dependence on functionality. (I.e. if the EvB is configured for Normal data taking, it cannot take low rate calibration data, and vis versa)
    - Normal running EVB flow settings
    - Calibration running EVB flow settings
- Due to use of non-guaranteed data transmission protocols, message data packets are lost.
  - Only the data flow has designed in data re-transmission
  - No such re-transmission hand shake for the message passing
  - If one allows too much data to flow through the ATM switch, messages are lost and the EVB breaks.

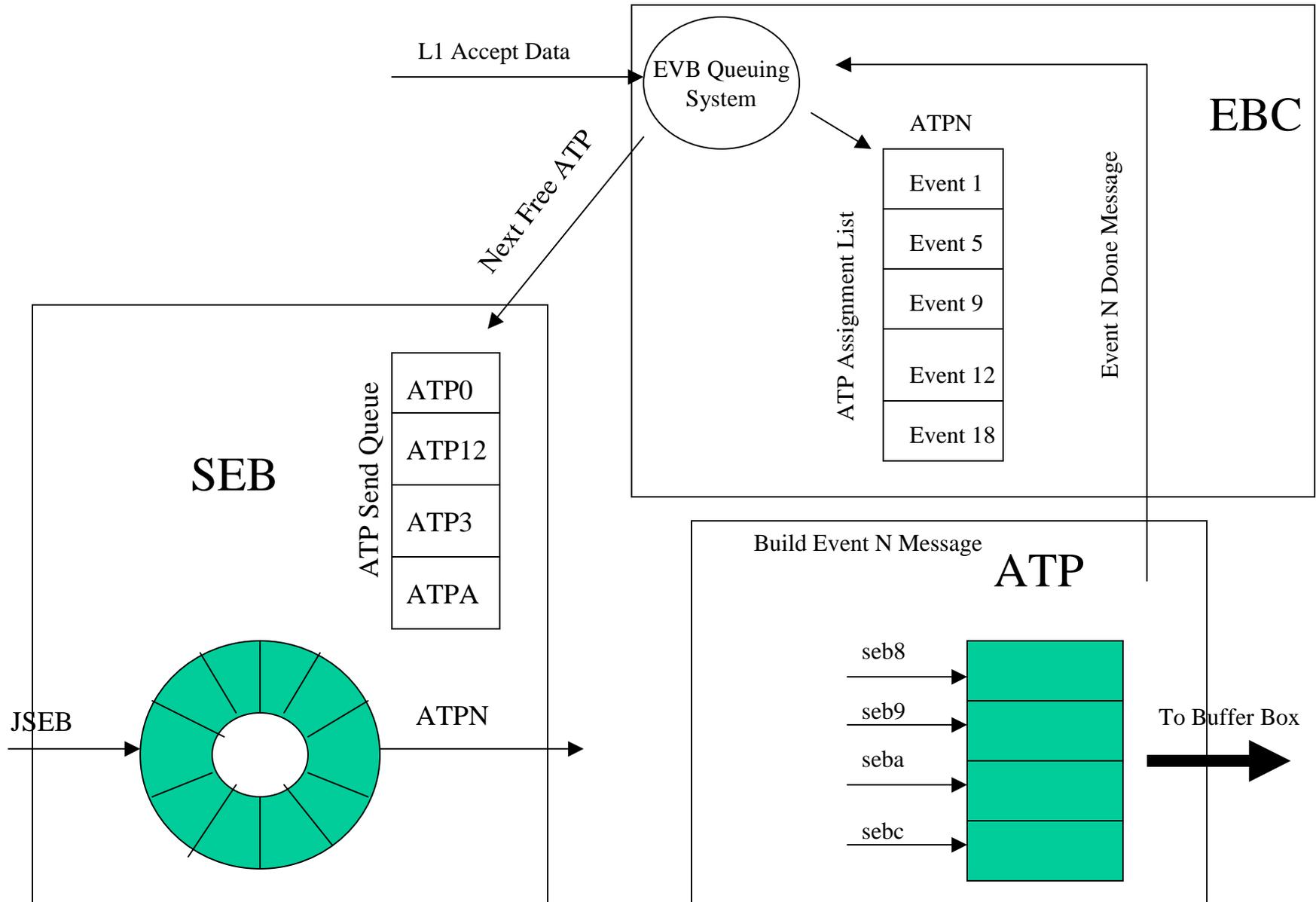
# Revisiting the Event Builder

- Assume we use only the Gigabit switch for the communication fabric.
- Rely on TCP for data transmission which has built in data integrity checking.
  - No need for home grown data re-transmission protocols.
- Use Gigabit switch for both data flow and message passing
  - Save on cost
- Rely on Unix based threading/ipc system to optimize event builder operation.
  - Run offline code directly in level 2?
- Keep option open for separate communication fabric for the messaging.
  - Dedicated 100BaseT, USB, Reflective Memories, Arcnet?

# A TCP Driven Event Builder



# Event Building Connect-o-gram



## Event Building Connect-o-gram Notes

- SEB output to ATP's should be multi-threaded.
  - one thread per socket connection
- ATP input connection from SEB's is also multi threaded.
  - Data can be dumped into ring buffers, one per SEB
  - Assembly thread
    - Checks to make sure all sub events have arrived
    - Merge the sub events
    - Frees up sub events in ring buffer
- No need for flow control
  - Event builder should run regardless of the Level 1 data rates.

## Further Notes

- We live in a predetermined DAQ environment
  - All sub systems will send the same number of sub events as the number of Level 1 accepts issued by GL1
    - If not, we have big problems and the DAQ should be stopped and problems investigated.
- Thus one can design the event builder system to take advantage of this assumption.
  - Let the underlying data transfer protocol take care of pack re-transmission.
    - Assume that all data sent over the socket will make it to the other end.
- No need to build in retransmission protocols or worry about “flow control”.

# TCP Preliminary Data

- A preliminary test was performed testing message passing rates over TCP
  - At what rate can one send small data packets
  - Can they be sent over the same fabric which carries the data
  - How much is the CPU taxed
- Setup:
  - Dual 2.4GHz Intel Xeon system
  - Intel 82544EI Gigabit Ethernet Controller
  - Gigabyte memory
  - Network Foundry gigabit switch

# The Test

- 2 types of tests were performed
  - Data through put
  - Message passing rate
- Data through put
  - TCP data transfer rate – 118Mbytes/sec
    - CPU Utilization 65% receive end/35% send end
    - No data packet loss
  - UDP data transfer rate – 118Mbytes/sec
    - CPU Utilization 30% receive end/30% send end
- Message Passing
  - TCP Message passing rate
    - 56KHz (128 bytes message buffer size)
    - 45KHz (128 bytes message buffer size) both ways
    - 20Khz (receive)/19Khz (send) with 113Mbytes/sec transfer on going

# Further Test

- Test of the Foundry Gigabit Switch
  - 10 + 10 systems sending data to each other achieve greater than 1 gigabyte/sec aggregate data transfer rate

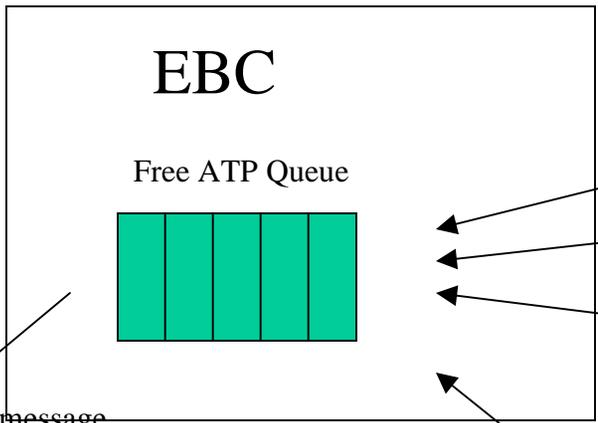
# Post Tests

- Hardware
  - NIH/CIT/DCB PET imaging development cluster
    - Dual AMD 1500Mhz (clock)
    - Gigabyte of main memory
    - 100BaseT 3Com NIC
    - HP 100BaseT Switch
    - 24 nodes
- Software
  - MPI
    - MPICH MPI 1 implementation software
      - Developed at Argon National Laboratory
    - Used to co-ordinate the startup and shutdown of all the nodes.
      - Uses ssh mechanism (No CORBA)
  - Home Grown multi-threaded queuing system.
    - Allows for N-buffering data/messaging scheme
  - Home Grown socket connection system.
    - Encapsulates all the TCP socket setup detail

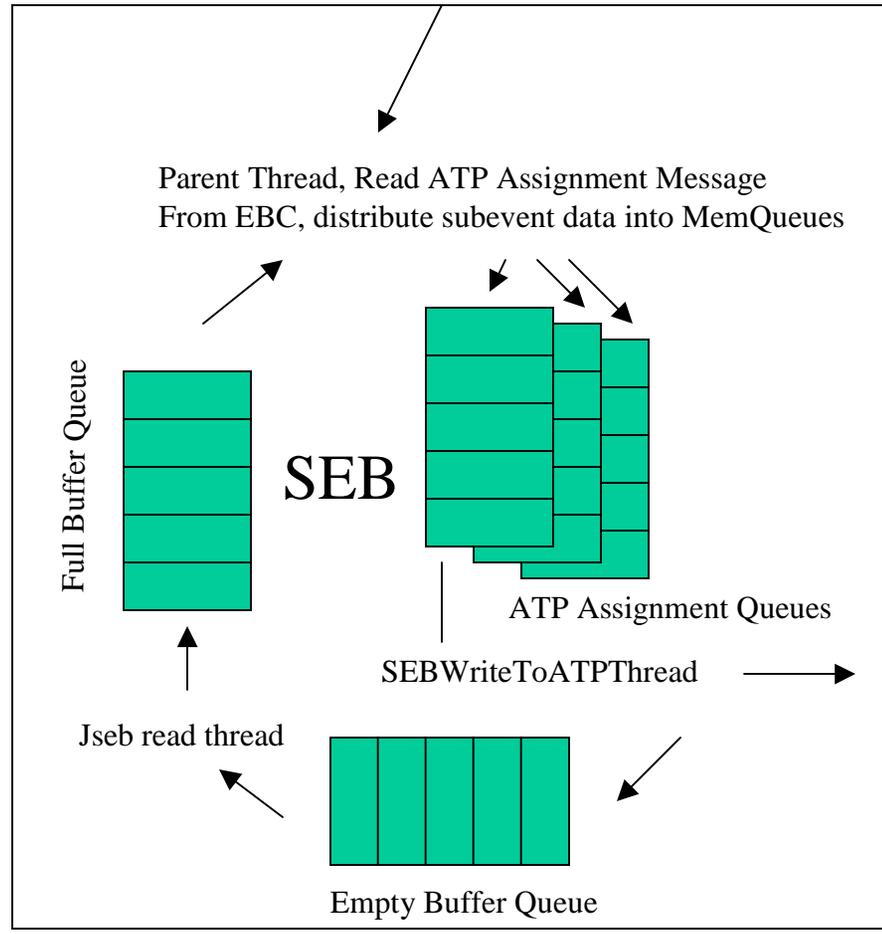
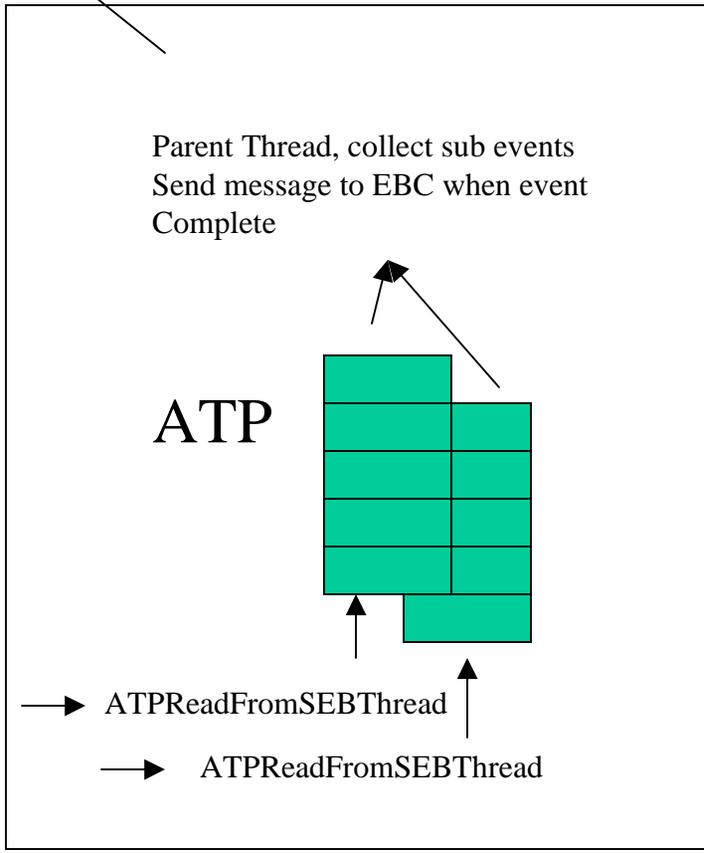
## More on the Queuing system

- Based on a get/put read/write scheme
  - Pushing data onto the queue you call 2 routines.
    - GetWriteBuffer()/PutWriteBuffer()
      - This grabs an element from the queue and you own it.
      - Any kind of element, a buffer pointer, a structure, reference to an object.
      - After using it, (writing into the structure,) you put it back into queue. (ring buffer?)
    - Pulling data off the queue you call 2 routines
      - GetReadBuffer()/PutReadBuffer()
        - Again, grabs element from the queue, (ring?) you can do something with the element.
        - When done, you put it back into the queue, (ring)
    - Multi-threaded.
      - Many threads can push data into the queue
      - Many threads can read data from the queue
      - First come/first serve basis
        - Multi-threading queuing is handled by mutex lock/unlock implementation

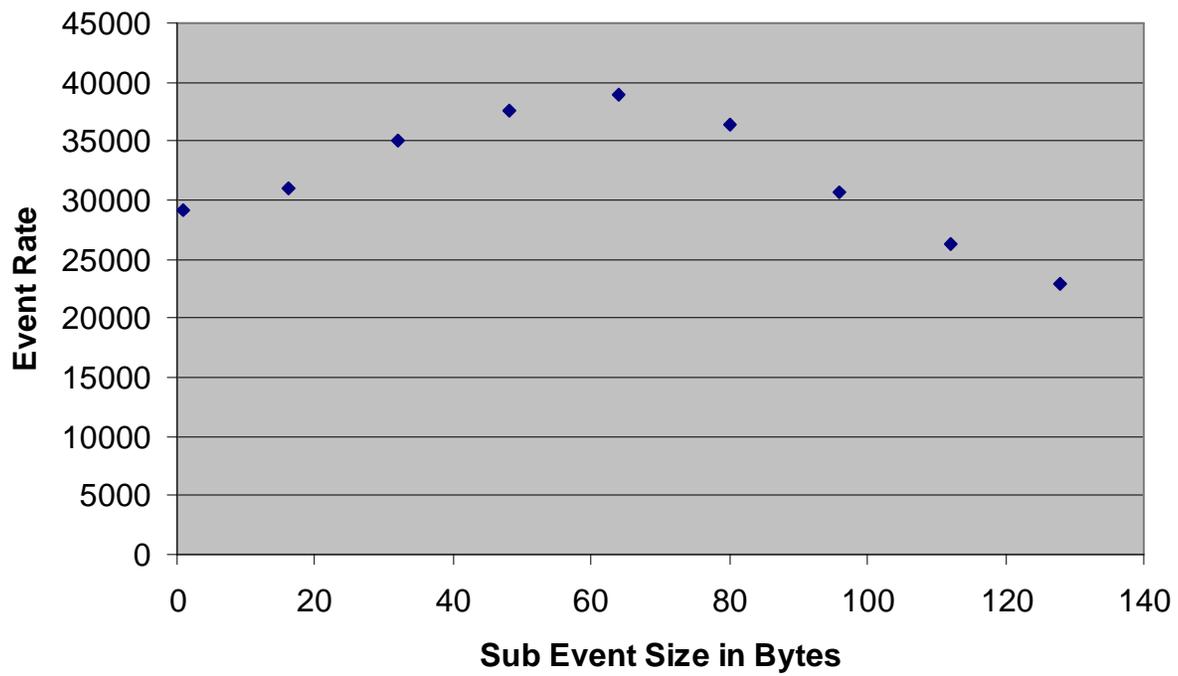
Very crude  
EVB system  
diagram



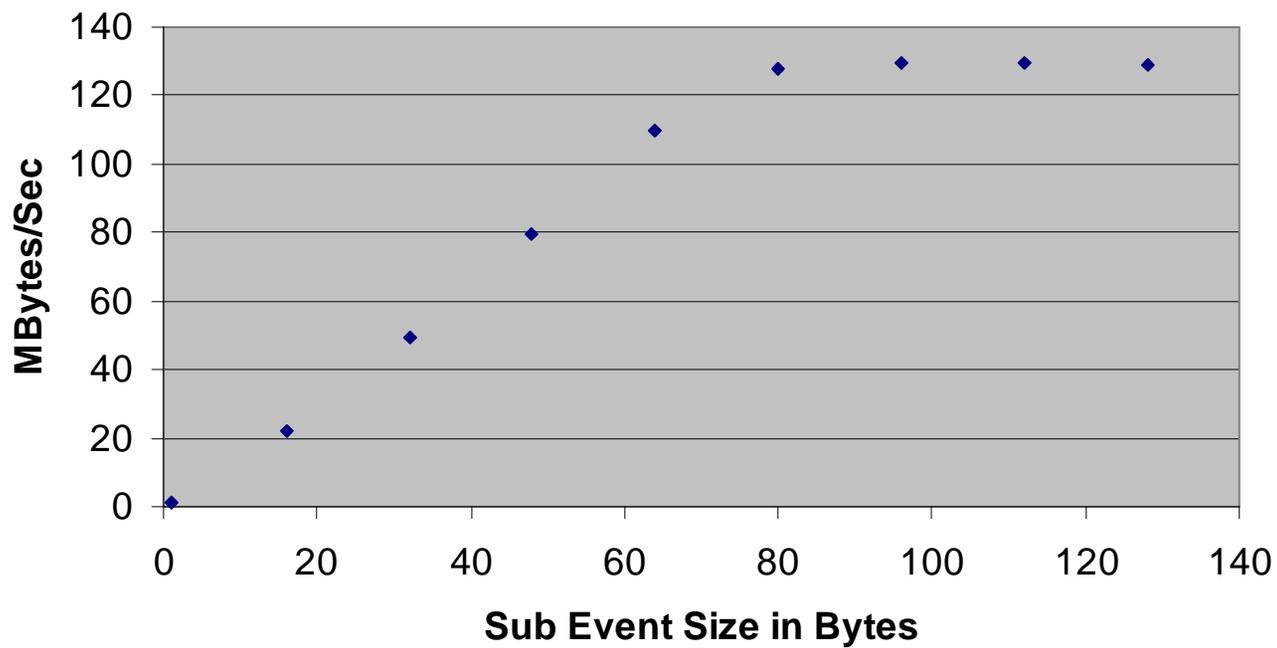
Other ATPs  
Other ATPs  
Other ATPs



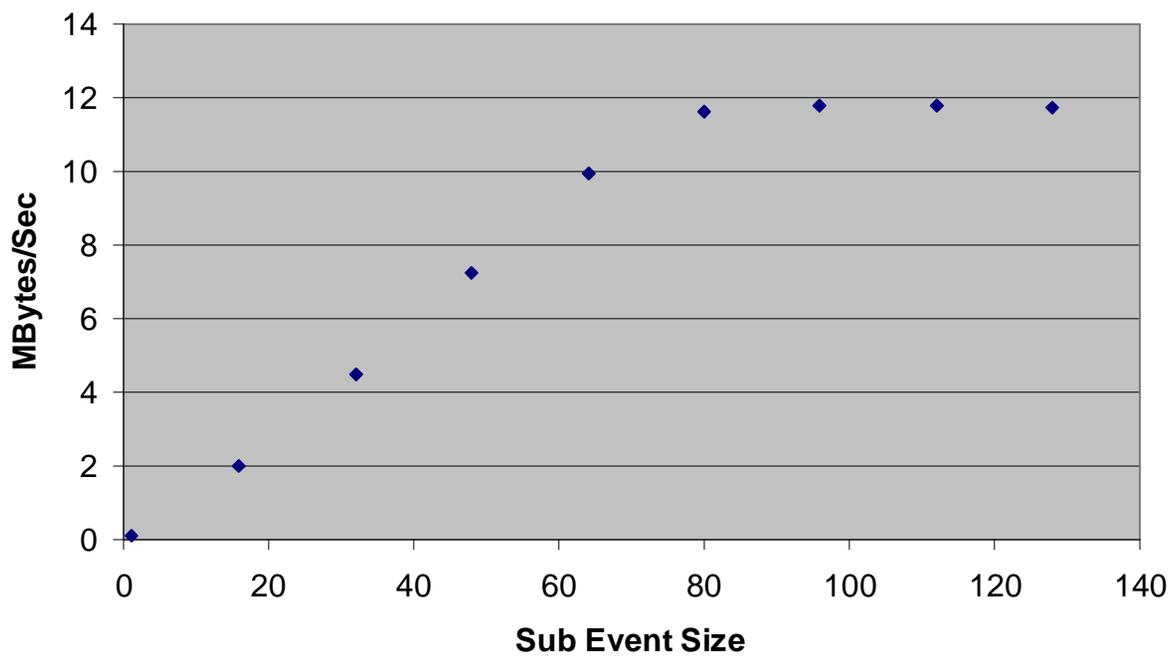
### Event Rate



### Aggragate Data Rate



### SEB Data Rate



# Conclusion

- The Event Builder plays a critical roll in the overall DAQ system.
- Currently there are problems which are keeping it from running at design rates, and thus limiting the Physics reach of the PHENIX experiment.
- It's time to start looking into the innards of the Event Builder think of new paradigms which will help speed it up.
- A TCP/Linux based event builder system is one such new paradigm.
- Critical eye must be kept on the messaging rate of the event builder.
  - Additional hardware dedicated to speed up the rate of message passing should be considered.
    - Dedicated 100BaseT network, USB, reflective memories...
- Prototype work should commence using a small number of nodes.
  - Test the feasibility of new ideas.
  - measure scalability.