Jefferson Lab

The ET System – High Speed Event Transfer and Distribution via Shared Memory and Networks

Elliott Wolin, Carl Timmer, D Abbott, V Gyurjyan, G Heyes, E Jastrzembski

Thomas Jefferson National Accelerator Facility, Newport News, VA 23606

Introduction

The ET (Event Transfer) system transfers high-volume, high-speed experimental data (events) from process to process within a single computer while also working seamlessly across a network.

ET is designed for use is in high-speed DAQ systems, where data typically moves in a single direction, from front-end hardware to permanent storage. Along the way the data is usually manipulated or modified, often by a chain of processes on a single node.

The ET system is quite general, and can be used in any application that involves transfer of data between threads/processes.

ET Data Transfer Model

Data is stored in shared memory, and transfer of data from one process to another on a single node is via transfer of a simple pointer into the shared memory, so no data is copied. Remote nodes access ET data via TCP/IP, and data is only copied when necessary (e.g. a monitoring process may only need a read-only copy, whereas other processes may modify the data and send it back).

The ET API is independent of whether shared memory or network transfer is used, allowing for run-time determination of the data transfer architecture.

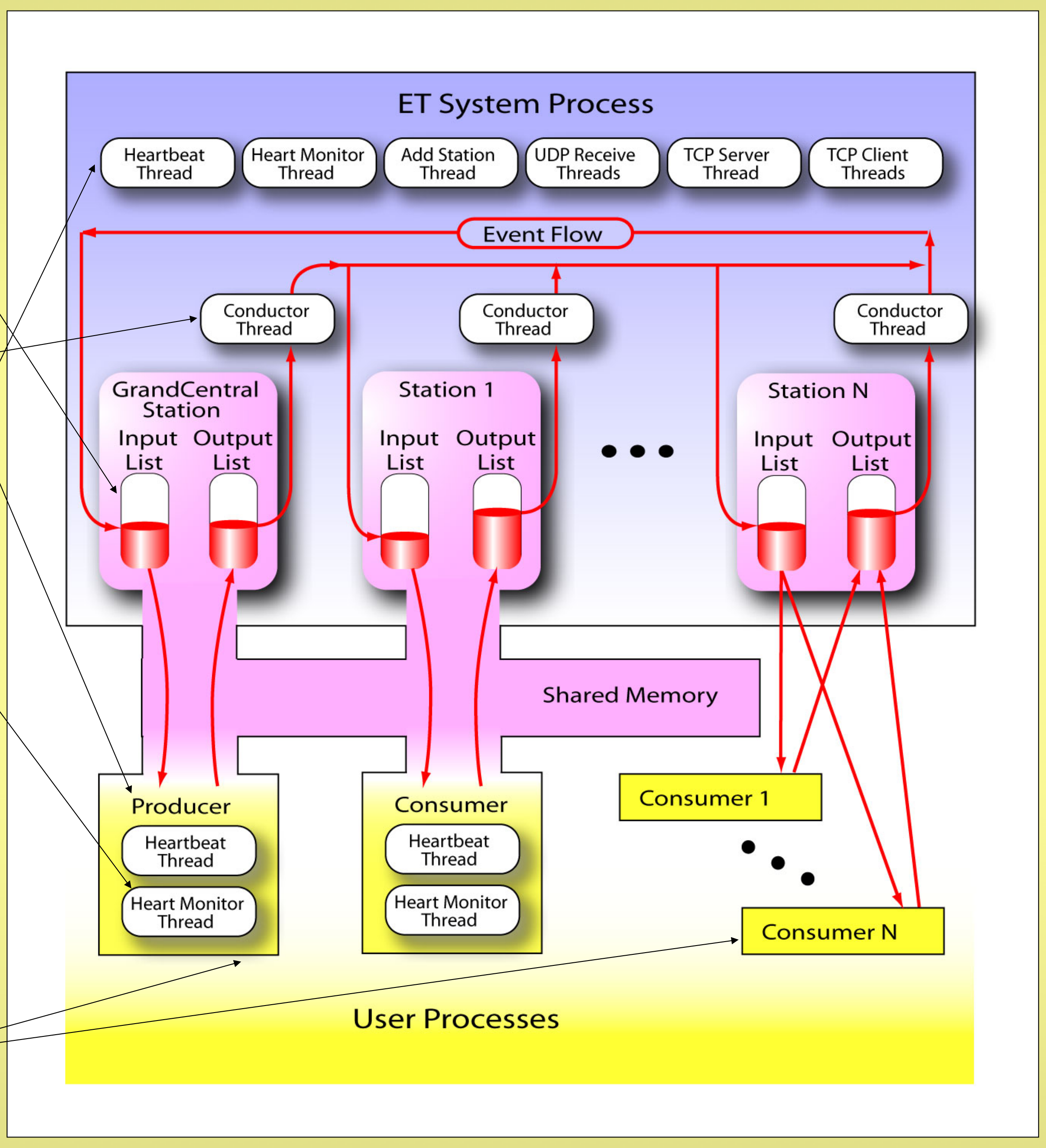
List of pointers to buffers (events) available for use.

Producer of data grabs some events, fills them with data and puts them back.

A conductor thread takes the producer's events and puts them into the next station's input list.

The ET system and local clients have threads that monitor each other's heartbeats and know if they're dead or alive.

ET clients can use shared memory if local or they can (transparently) use the network to get and put events.



Events have settable header fields which can be used to hold metadata. Once events are past the last station, they are returned to the pool at GrandCentral station to be used again.

Stations of various configurations can be added at runtime. There are many different configuration options. For example, they can be configured to filter events on user criteria.

Stations are shown here in series. They can also be placed in parallel. When parallel, events can be distributed among such stations in round robin fashion, or they can be distributed to ensure equally occupied input lists for load balancing.

ET clients can discover ET systems by broadcasting and/or multicasting in order to connect to it, as well as connecting directly.

Advantages

- **Runs on Linux & Solaris with client library on vxWorks**
- **Java version available which interoperates with C version**
- **Java monitoring GUI**
- **Code is reentrant (run multiple copies on one machine)**
- **Overhead for handling events is negligible**
- **Speed limited only by network bandwidth and machine memory copy speed**

Conclusions

The ET system implements high-speed data sharing among processes on the same node via shared memory, and high-speed data transfer between processes on separate nodes. Data movement is minimized whenever possible.

A rich variety of connection, selection, and transfer modes is implemented, allowing for the extensive tailoring and customization required by state-of-the-art DAQ systems. The ET system API is fairly simple and straightforward, allowing non-experts to use ET.

The ET system has formed the foundation for data transfer and sharing for all JLab high-speed DAQ systems for many years, and will continue to do so at the upgraded 12 GeV facility at JLab.

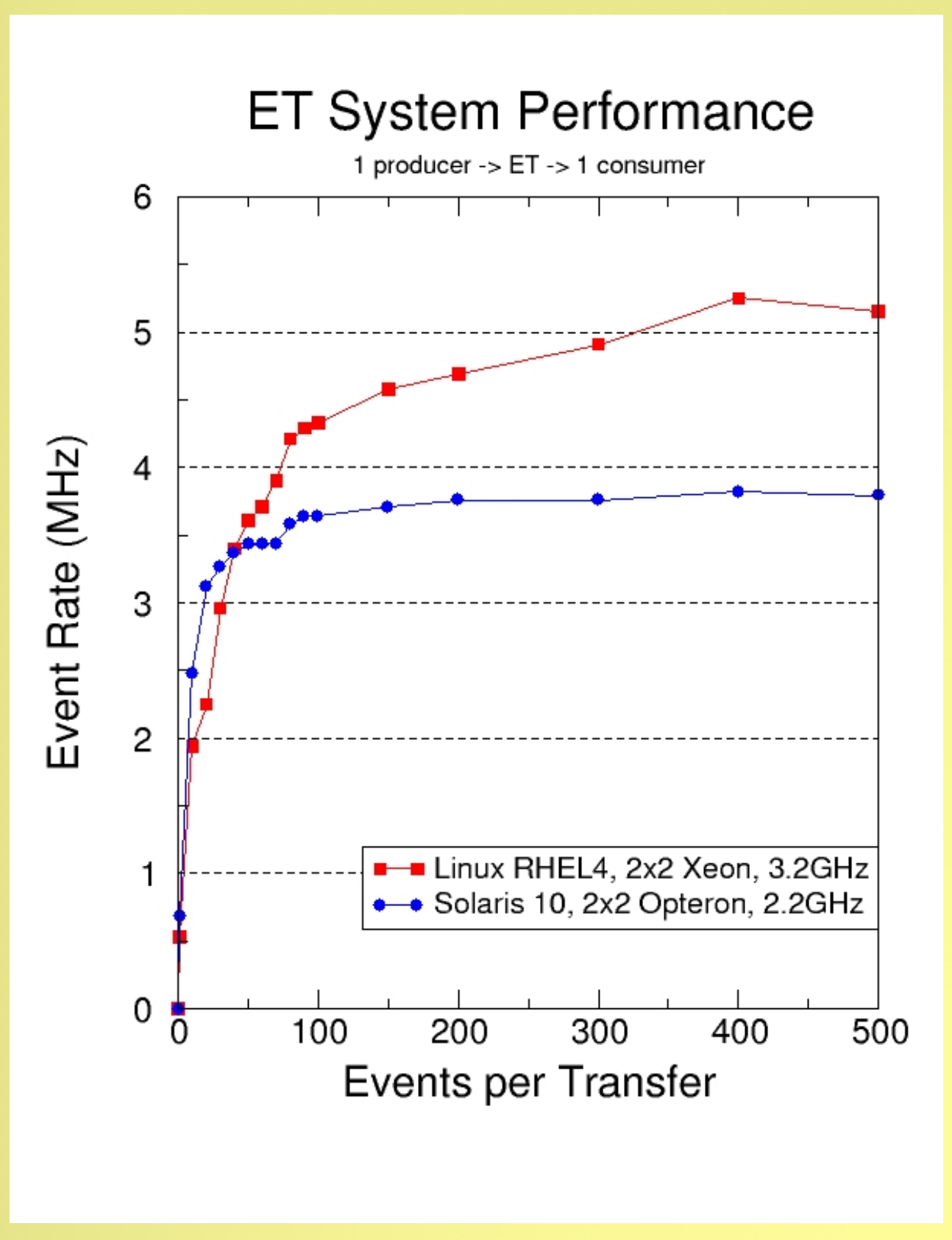
Downloads

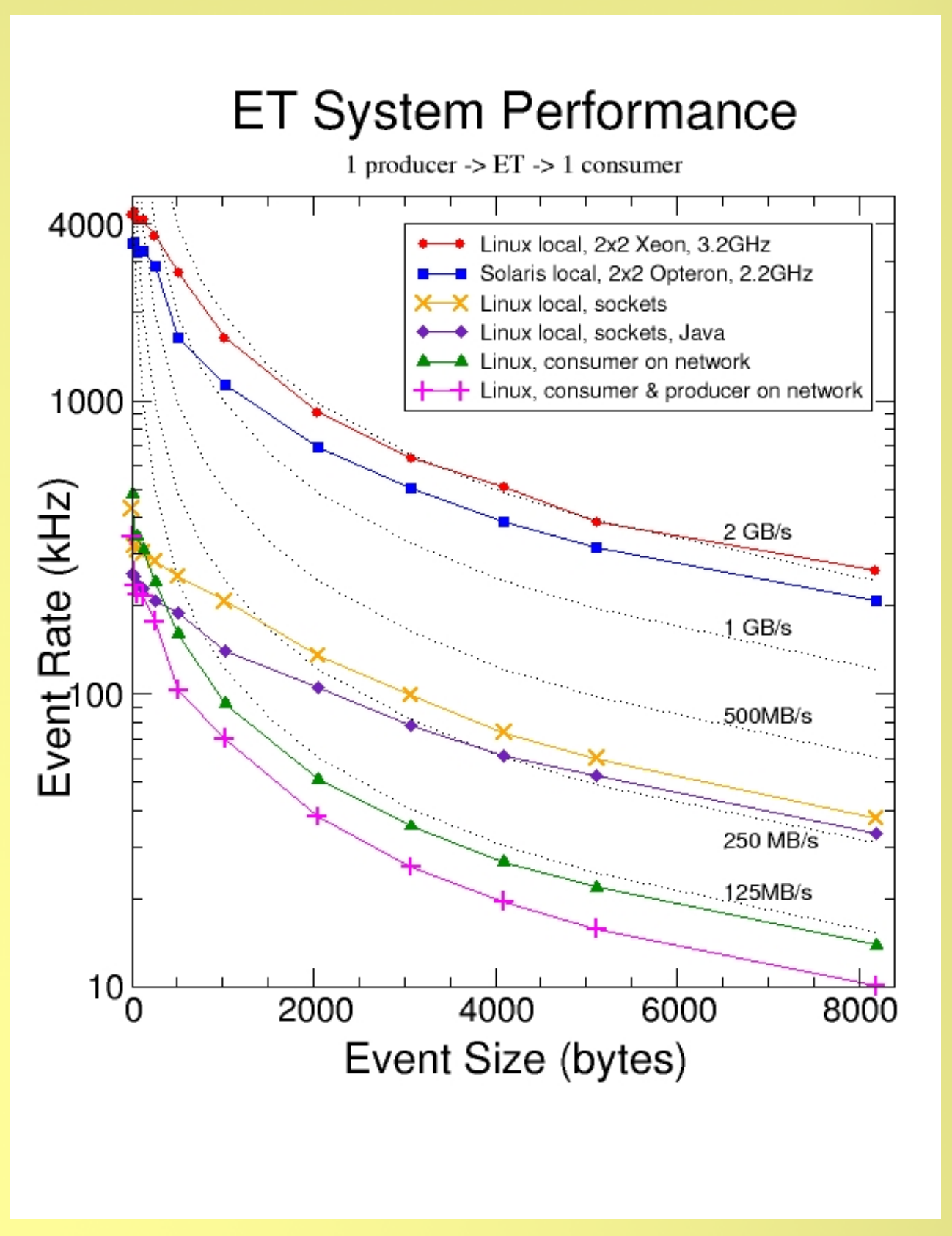
Download and give ET a try! You can get your free copy today at ftp://ftp.jlab.org/pub/coda/ET:

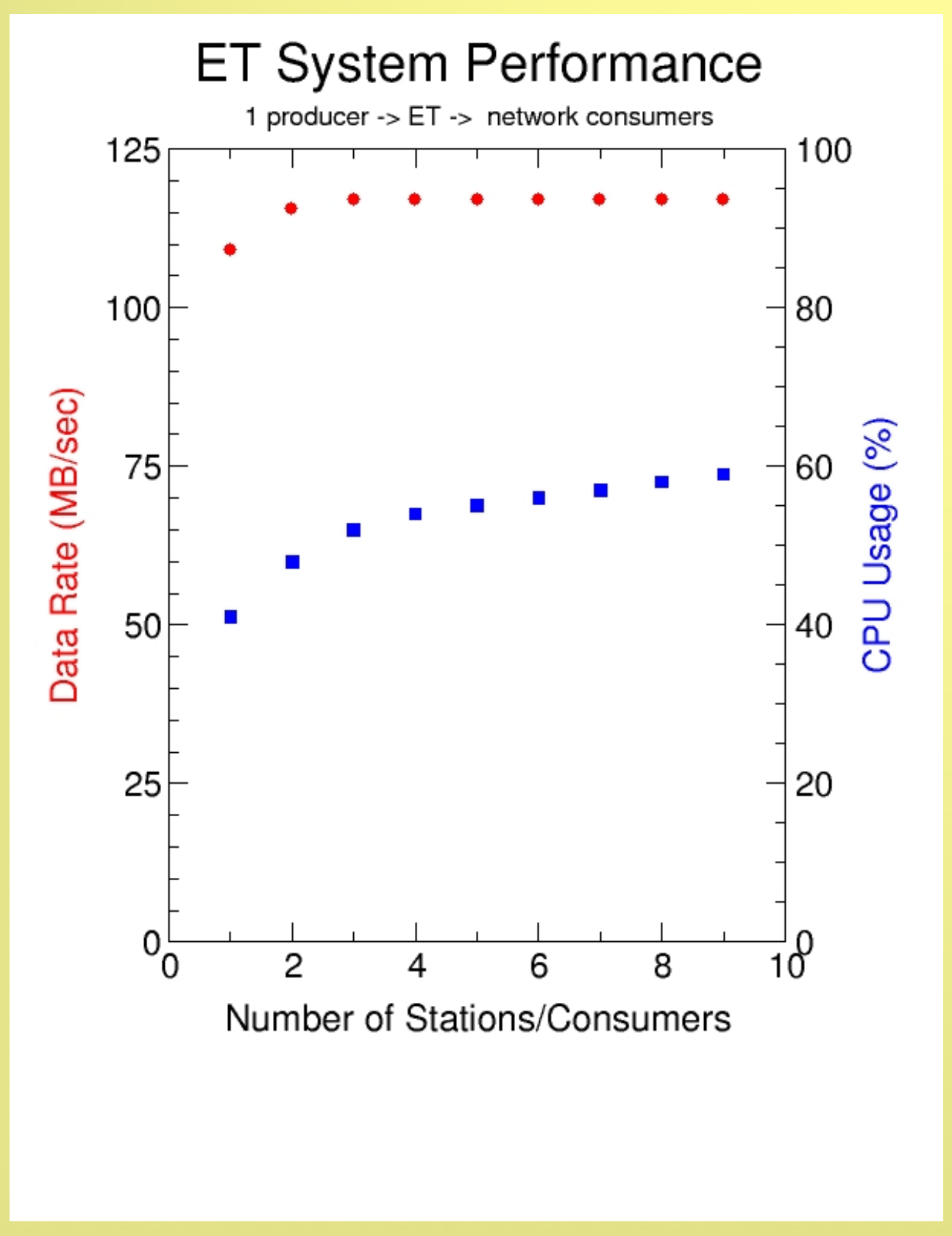
Carl Timmer, (757) 269-5130, timmer@jlab.org, or

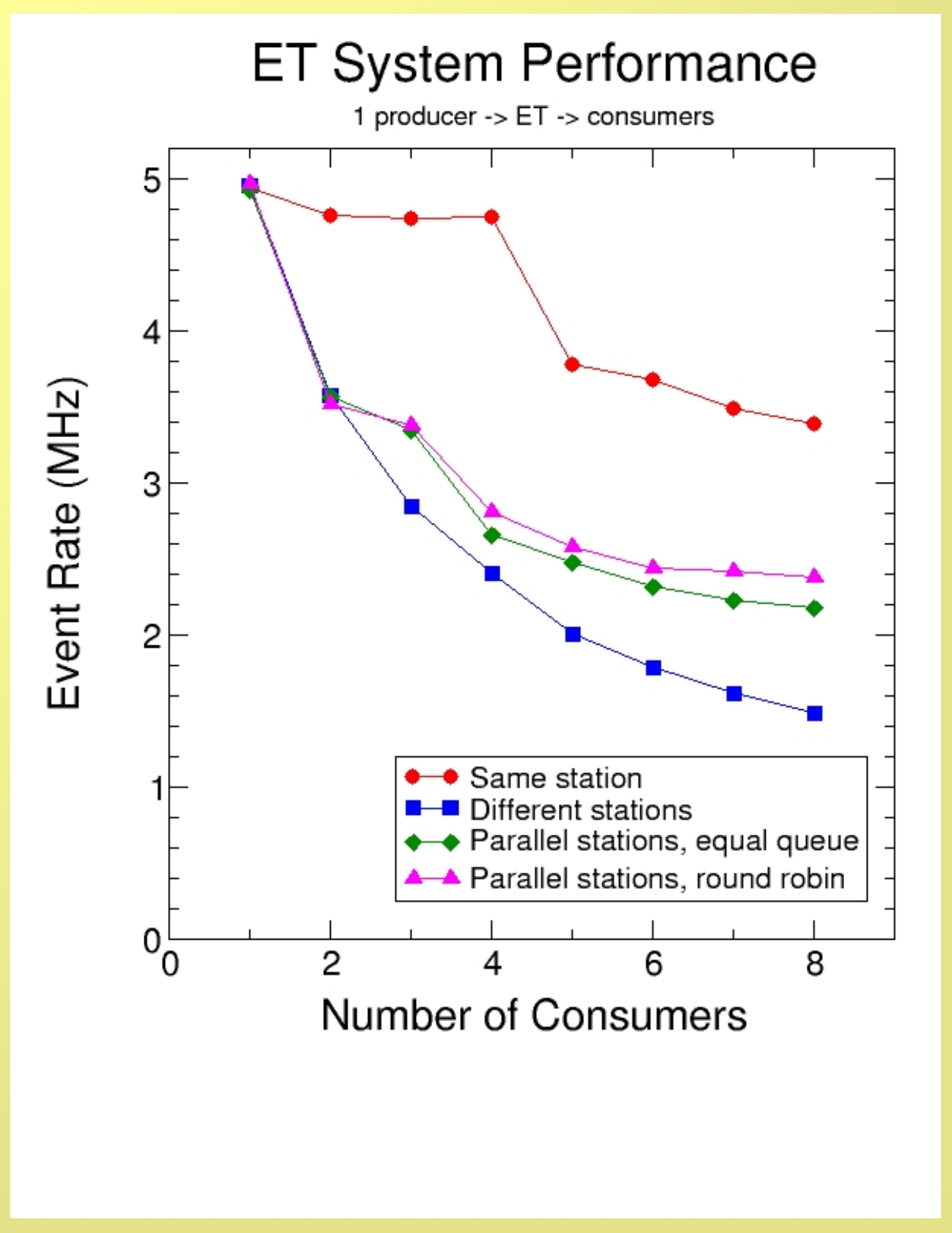
Elliott Wolin, (757) 269-7365, wolin@jlab.org

Performance


ET System Performance
1 producer -> ET -> 1 consumer
Event Rate (MHz)
Events per Transfer
Linux RHEL4, 2x2 Xeon, 3.2GHz
Solaris 10, 2x2 Opteron, 2.2GHz


ET System Performance
1 producer -> ET -> 1 consumer
Event Rate (kHz)
Event Size (bytes)
Linux local, 2x2 Xeon, 3.2GHz
Solaris local, 2x2 Opteron, 2.2GHz
Linux local, sockets
Linux local, sockets, Java
Linux, consumer on network
Linux, consumer & producer on network
2 GB/s
1 GB/s
500MB/s
250 MB/s
125MB/s


ET System Performance
1 producer -> ET -> network consumers
Data Rate (MB/sec)
CPU Usage (%)
Number of Stations/Consumers


ET System Performance
1 producer -> ET -> consumers
Event Rate (MHz)
Number of Consumers
Same station
Different stations
Parallel stations, equal queue
Parallel stations, round robin