FUNNEL: TOWARDS COMFORTABLE EVENT PROCESSING

Burkhard D. Burrow\textsuperscript{a,b}

Universität Hamburg, H. Institut für Experimentalphysik,
Luruper Chaussee 149, 22761 Hamburg, Germany

The funnel software package has solved the ZEUS collaboration the problem of Monte Carlo event production. A problem faced by many HEP experiments. Thanks to extensive automation, a few man-hours per day are sufficient to resolve problems and to manage the entire ZEUS Monte Carlo production. Other than specifying the events to be produced, ZEUS physicists are thus freed from the chore of Monte Carlo production. As an additional benefit, the computing cycles required for production are nearly cost free since they replace otherwise idle cycles on hundreds of Unix workstation and server computers, with minimal interference for their regular users. The computers are spread across a dozen sites around the world and continually deliver the effective equivalent of approximately one hundred dedicated computers.

Funnel successfully demonstrates that generic independent tools can provide comfortable event processing. With an emphasis on automation and fault-tolerance, the tools manage all aspects of event processing including the job queues, the execution and failures of the processing program, parallel processing, as well as data buffering, archiving and remote transfer. The L3, HERMES and H1 collaborations are presently creating Monte Carlo production systems, using the funnel experience and, to different extents, parts of the funnel software package.

The experience gained with funnel encourages the construction of EVPRO, a general purpose software package for event processing. EVPRO would build on top of existing software; for example CPS or PVM for parallel processing. Whether on a dedicated farm of computers or using idle cycles, an application of any size could then easily enjoy the comfort of automated, fault-tolerant event processing. EVPRO aims to minimize application-specific event processing software, whose high development costs can only be justified for the largest of applications. A casual user may provide EVPRO with only the processing program and the data to be processed. A more complex or real-time application would tune EVPRO to its needs; for example, integrating custom hardware for the flow of event data. Making optimal use of the available computing resources, EVPRO would manage all aspects of the event processing. Monte Carlo production, event reconstruction and software triggers could use EVPRO, as could any computing application, inside or outside of HEP, which can be expressed in terms of events.

In principle, event processing is a solved or even a trivial problem. Given an implementor, EVPRO could provide the trivial solution in practice.

1 Introduction

The ZEUS experiment at DESY records order $10^7$ events per year at the HERA electron-proton collider. The physics analyses require several times more Monte Carlo (MC) events than HERA events; on average $5 \times 10^7$ MC events/year $= 10^8$/week $= 100$/minute. Due to the large number of analyses, the required MC events are distributed across approximately 100 job requests per week. The MC simulation of an event typically consumes one minute of workstation CPU. Thus, ZEUS MC production effectively requires

\textsuperscript{a} Presented in the plenary session 'E: Systems and Facilities' at Computing in High Energy Physics (CHEP95), Rio de Janeiro, Brazil, 18-22 September 1995.

\textsuperscript{b} Also supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by the BMBF under Contract Nr.056/HH291.
The continual use of 100 dedicated workstations. With an average MC event size of 60 Kbytes, the total production rate of 100 Kbyte/second can be carried by local area networks (LAN), though not generally on wide area networks (WAN). An aggregate production of about 60 Gbytes/week = 3 Tbytes/year must be archived.

1.1 Resources for ZEUS MC Production

There are approximately 500 physicists in the ZEUS collaboration, providing access to more than 500 workstations spread across sites in Europe, North America and Asia. Their otherwise idle CPU time is harnessed to effectively provide the 100 dedicated workstations required for MC production. At each site, the LAN connecting the 5 to 90 workstations allows MC production to use parallel processing. All the sites may be remotely controlled using rlogin/ftp/sh. The WAN, idle night bandwidth permitting, or Exabyte tapes sent by courier mail transport the produced events to the MC archive at DESY. The above resources are managed by funnel to provide the ZEUS MC facility.

1.2 The ZEUS Physicist’s View of Funnel

Physicists submit jobs to funnel. Each job provides the input events, in terms of particle momentum vectors, to be sent through the MC simulation specified by the job. The requested detector and trigger simulation and event reconstruction may range from one of the standard ZEUS configurations through to a physicist’s private development software.

Physicists receive from funnel for each job the archived output events, log files and information on any input events which crashed the simulation software.

ZEUS physicists are thus freed by funnel from the chore of MC production.

2 Global ZEUS MC Production

2.1 Transferring Gbytes Around the World

Each input job is sent over the network to one of the production sites for processing. To transfer data to a remote site, it is placed into a local transfer directory. This is an example of a daemon directory. It is monitored by a daemon process, which runs in the background and acts on every file appearing in the directory; logging the files, the actions and the exceptions handled. A successful action is noted by deleting the file or moving it elsewhere. Faults beyond the daemon’s exception handling are e-mailed to the user. The action performed by the daemon is thus asynchronous, modular, automated and fault-tolerant. Once a process places a file in a daemon directory, the file requires no further actions nor exception handling from the process; the daemon effectively guarantees success.

Depending on the site, the output data is transferred to the MC archive by the farc daemon, using Exabyte tapes, or by the fmvbat daemon, using ftp.

2.2 The Job Queue at Each Site

To have an input job processed, it is placed into the local job queue directory. The job queue daemon ensures that for each job, the output data appears in the archive or, if the job fails in a manner beyond the daemon’s exception handling abilities, e-mail notifies the funnel operator. For example, the daemon reverts a job aborted by a computer reboot and notifies the funnel operator if an input job is corrupt.
3 Comfortably Processing a Single Job

In an ideal world, with infinite computer speed, unlimited disk space and bug-free programs, processing a single job is trivial. The processing program is simply run on the input events in order to produce the output events.

In the real world, event processing is not trivial. A job may take days to complete and produce many Gigabytes of output, but neither time nor space are generally available. In addition, others may want to use the computer, the computer or disk or network may crash or for some of the input events, the simulation program may crash or enter an infinite loop.

The funnel tools can’t provide the ideal world but, as the following outlines of some of the major tools show, they can provide its comfort to good approximation.

3.1 Buffering Gigabytes of Output

The output data produced by a job is buffered on disk during processing. The buffering is performed automatically by the following two funnel tools. Once the job has completed, symbolic links effectively place the buffered data into a transfer directory of section 2.1.

The getBUFFER utility manages free disk space. In response to a process’ request, getBUFFER returns an address on a disk with free space greater than the current request plus outstanding allocations on that disk. Thus, the allocated space is guaranteed to be free for use by the process. getBUFFER automatically reclaims allocations outstanding to processes which have exited. getBUFFER does not delete ‘old’ data; that’s left up to the application. Thus one or more processes may easily use multiple disks.

The dogB utility places a process’ output into a getBUFFER area or areas. dogB uses Unix I/O to block the process until the getBUFFER area is available. Thus, the process is gracefully blocked while the disks are full, resuming once there is free space.

Thanks to getBUFFER and dogB, the process effectively enjoys unlimited disk space.

3.2 Managed Event Processing and Parallel Event Processing

The real world problems of job processing, those not related to disk space, are solved by making use of the natural unit of processing, the event. The input data consists of independent events, each of which is independently processed to produce an output event.

The solution keeps the simple original processing program, schematically shown in Figure 1a), but introduces the manager program as shown Figure 1b). The event is the manager’s unit of input and output (I/O). By manipulating entire individual events, the manager is generic across different processing programs and can solve the problems of event processing. For example, if using the idle cycles of a computer, the manager recognizes when other users become active and then terminates or suspends the processing program; resuming event processing once idle cycles are again available.

Bugs are inevitable in large computer codes such as the ZEUS simulation programs. While bugs are eventually fixed, they are often irrelevant and should thus not halt event processing. In analogy, HEP experiments don’t usually halt data taking just because an online program occasionally crashes. Therefore, if the processing program crashes, the current event and the stack trace are recorded by the manager, which then continues processing. As described in section 1.2, a job’s output data is accompanied by the crash information. The physicist can thus determine the relevance, if any, of a crash and can pursue the code’s author(s). Similarly, if the processing program enters an infinite loop, the
manager e-mails the author(s), who can then examine the running loop with a debugger. The processing program is eventually forced to crash and is treated as described above.

The independent events may obviously be distributed1 across many copies of the processing program. As illustrated in Figure 1c), the para-manager program easily implements this trivial parallelism. As for the manager program, by manipulating entire events, the para-manager is generic across any processing program. Intrinsically simple, the para-manager provides various features such as the dynamic connection and disconnection of clients, allowing the use of idle cycles for processing.

4 EVPRO for Comfortable Event Processing

For event processing, be it MC production, a software trigger or event reconstruction, HEP is concerned with the input and output events, the processing software and any exceptions. HEP has no intrinsic interest in managing the computer resources passing the events through the processing software. As outlined in sections 2 and 3, the funnel architecture demonstrates that generic, independent tools can manage the computing resources required for event processing. Going beyond ZEUS, funnel thus encourages the construction of EVPRO, a general purpose software package for event processing. EVPRO could benefit HEP tremendously, since event processing is currently managed manually and/or using application-specific software. This conference alone provides many examples3 in addition to funnel, of the HEP energy currently spent managing event processing; energy to be spared by EVPRO. Interest in EVPRO is corroborated by the L3, H1 and HERMES interest in funnel mentioned in the abstract. At present, EVPRO does not exist and seeks an implementor(s).

EVPRO would consist of event processing tools, containing some new software, and information helping the user determine the solution to the event processing problem. The solution may involve user, EVPRO and other software. The other software, which EVPRO would also heavily use, could include for example CPS or PVM for parallel processing. CERN’s SHIFT utility for disk space management and NQS or LSF for job management.
Passing a file of events through a simulation program demonstrates a simple use of EVPRO. After relinking the simulation program to use the EVPRO manager’s event reading and writing routines and providing a routine identifying the input data event boundaries, the physicist would be freed of the details of the event processing, including parallel processing, disk management and most exception handling. For many such jobs per day, EVPRO would provide the tools required for a funnel-like MC production facility.

HEP software triggers using distributed event processing could be managed by EVPRO as illustrated in Figure 2a). The hardware for event I/O is integrated into the manager program which shares event memory with the trigger program.

Outside of HEP, many applications are similarly divisible into events or tasks, for which, the calculation time is greater than the data transfer time. As shown in Figure 2b), EVPRO could provide comfortable master-worker parallelism to these applications.

![Figure 2](image)

**Figure 2** a) EVPRO for a HEP software trigger. b) EVPRO for generic master-worker parallelism.

**Summary**

Freed by funnel from the chore of MC event processing, ZEUS physicists have more time for physics. In addition, funnel provides inexpensive MC production using idle cycles.

Implemented as EVPRO, funnel’s idea of generic independent event processing tools should be portable to HEP and other applications. Freed by EVPRO from the chore of event processing, HEP should then have more energy for physics.

**Acknowledgments**

Marc St-Laurent created the fare tape daemon. Wouter Verkerke created the software managing the global flow of jobs. ZEUS MC production and the funnel software owe much to F. Benard, L.W. Hung, L. Wai, R. Yoshida, J. Butterworth, M. Kuze, D. Gikins, A. Bruni, M. Botje and H. Uijterwaal. Strong support comes from the DESY computing services and from the many ZEUS colleagues at the production sites and in the ZEUS software groups. Special thanks to ZEUS for all its otherwise idle computing cycles.

**References**

1. The HEP precursors of techniques such as data transport by tape and event farming are described in the proceedings of previous CHEP conferences.
2. S.W. O’Neale, CHEP92, p.471, describes a similar service for OPAL.