PEP: the Phenix Event Processor

Soren P. Sorensen, Han Geng, Qihang Liu, David P. Morrison, Li Qun, Victor Perevoztchikov, Kyle Pope, Mark Pollack, and Yan Zhao

University of Tennessee, Knoxville, TN 37996-1200, USA

In order to be able to handle the large computing requirements of the PHENIX detector a fully distributed event processing application: PEP (The Phenix Event Processor) has been developed. PEP consists of a large set of individual processes distributed over a heterogenous environment communicating via message passing.

Key words: Massively Parallel Systems, Distributed Computing

1 Introduction

The PHENIX experiment at the Relativistic Heavy Ion Collider (RHIC) is scheduled to begin collecting data in the Fall of 1999. The very large off-line computing requirements are described elsewhere in these proceedings [1]. PHENIX will collect 60-70 events/sec during 40 weeks each year. For the most interesting, central Au+Au events the reconstruction CPU time is more than 1 GFlops-sec. The only way of being able to handle this large CPU requirement is to create a large-scale, distributed system. In PHENIX we have selected to design and implement a distributed system from the very begining in order to gain some experience on what kind of problems will be encountered running in a distributed, heterogeneous environment in contrast to a co-located, local environment.

2 Basic Components of PEP

The PHENIX Event Processor (PEP) is intended as the main tool for all event processing and data analysis within the PHENIX collaboration. The core part of PEP is completely free of any references to PHENIX and only the user supplied Data Operators will contain PHENIX specific "physics" software.

As illustrated in Fig. 1 PEP contains the following components:

Controler: This is the "brain" of PEP, which controls all other processes within a PEP session. Specifically the controller starts and stops all other processes, it controls all communication between processes and monitors the state of each process. Based on this information it can perform resource allocation (Object Brokering).

Graphical User Interface (GUI): The user interface is implemented with Tcl/TK and provides all interaction between the user and PEP at run time.

Data Operators are the "worker bee's" of PEP. They accept data structures as input, perform operations on them and provide data structures as output. All data structures are implemted using the DSPACK data structure package from University of Birmingham [2].

Input & Output Tool provides for all I/O of data structures and thereby screening the data operators from the actual physical device. Typical Input sources will be files, shared memory or the Online Data Acquisition System.

Data Base Tool provides remote worldwide access to the PHENIX's central ORACLE database at RHIC.

Visualizer provides all graphics services like histogram display, event display and graphical data interaction. In addition it provides monitoring of all PHENIX specific aspects of the data flow.

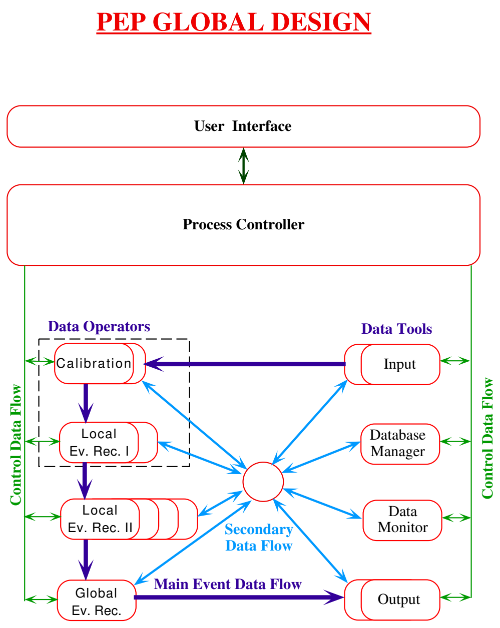

Fig. 1. An illustration of the main components in the first version of the PHENIX Event Processor. Please refer to the main text for further explanations.

3 Interprocess Communication

All interprocess communication within PEP is done by message passing. For a co-located system this is more cumbersome than using shared memory, but the great advantage of message passing is it's generality. Once a system has been implemented completely with message passing communication then it can be used on a large range of platforms and implemented with a large range of sizes. In particular PEP has been implemented so the controller only receives very small control messages, whereas the "heavy traffic" of physics data structures is only taking place between the numerous data operators and/or the I/O tools. Currently we have chosen to use PVM (Parallel Virtual Machine) to implement the message passing, since it makes it possible to implement PEP in a heterogeneous platform environment. We are also investigating if a CORBA implementation might be advantageous for the control messages.

4 Organization of Physics Modules

An important aspect of PEP is the functional organization of all "Physics" software into modules for each sub-detector part of the complete PHENIX detector. In general each subdetector has to provide the modules described below (with a few exceptions for Event Generators and Global modules). The interrelationship between the modules is shown in Fig. 2

Event Generator (Gen): Generates simulated events in the form of a list of particles created at the event vertex.

Hit Generator (Hit): Accepts the Gen output and generates hits in all sensitive volumes (based on GEANT).

Signal Generator (Sig): Accepts the Hit output, sums hits over all readout volumes and calculates the final physics output signal in each read-out channel.

Simulator (Fee): Simulates the response of the front-end electronics to provide the digital signals in each channel. Typical effects: digitization, discriminator walk, pedestals etc.

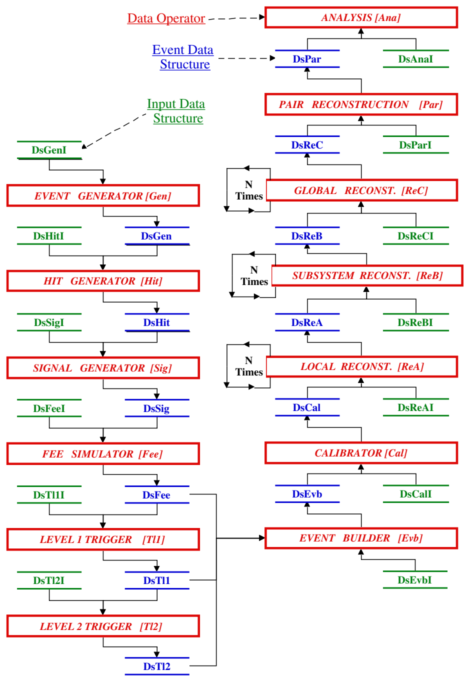

Fig. 2. The inter-relationship between all physics modules within PEP

Level 1,2,3 Trigger Simulator: Simulates all aspects of the data handling in the level 1,2 or 3 trigger systems.

Calibrator (Cal): Accepts the raw data as input an either calculates calibrations constants or performs all calibrations and corrections of the raw data.

Local Event Reconstruction (ReA): Performs all event reconstruction at the detector level which an be done in a stand-alone mode. Examples: Cluster finding in EMC, Hit finding in the Drift Chambers.

Sub-system Event Reconstruction (ReB): Performs all event reconstruction at the sub-system level, which is done based on the input of information from another detector in the same sub-system.

Global Event Reconstruction (ReC): Performs all event reconstruction at the global PHENIX level, where information from other sus-systems is necessary.

Analyser (Ana): Performs subsequent data analysis.

5 Acknowledgments

This work was supported by the Department of Energy through the RHIC project and through contract DOE DE-FG02-96ER40982.

References

[1] David P. Morrison, Han Geng, Qihang Liu, Li Qun, Mark Pollack, Victor Perevoztchikov, Soren P. Sorensen, and Yan Zhao: *The Off-line Computing System for the PHENIX Experiment at RHIC*, in the proceedings of this conference.

[2] R. Zybert and P. Buncic, *DSPACK - Object Manager for High Energy Physics*, Proc. CHEP95, p. 345