The PHENIX Experiment in the RHIC Run 7

Martin L. Purschke, for the PHENIX Collaboration
Brookhaven National Laboratory, Upton, NY 11973, USA

Abstract.
PHENIX [1] is one of two large experiments at Brookhaven National Laboratory's Relativistic Heavy Ion Collider (RHIC). Previously, the PHENIX experiment had completed very successful Runs that featured different beam species in RHIC. Since the previous Au+Au run in 2004, we had improved the performance of the data acquisition system by introducing several key technologies, such as distributed compression and multi-event buffering, that had increased the event rate to 5KHz for the smaller collision systems that we measured since, Cu+Cu, and p+p. However, this rate had yet to be reached in the largest and most demanding collision system, Au+Au at 200AGeV.

Previously, the data logging capability had been demonstrated to exceed 600MB/s, although this value had never reached in actual running. The RHIC luminosity and the smaller collision systems before did not deliver enough data to saturate the logging capability. In this Run 7, we exceeded the 600MB/s logging rate for the highest multiplicities, where we also reached the current practical limit of 5KHz event rate with the large events size.

We have in the past used GRID middleware tools to transfer substantial data volumes in the order of 400TB off-site in order to make use of remote computing capacity in Japan, France, and other sites in the US, and used this technology to achieve priority processing of triggered events again in Run 7.

The Run 7 also introduced several new prototype detectors in the experiment, a new high-resolution Time-of-Flight detector, a reaction plane detector, a Muon Piston Calorimeter, and, as a prototype, a Hadron Blind Detector that is designed to tag the background from conversion electrons and improve the photon and electron measurement capability of the PHENIX apparatus

We will describe the ingredients which made those highest rates possible, and show some performance figures.

1. Introduction

PHENIX [1] consists of 4 large spectrometer arms, two central arms and two forward Muon arms (Fig. 1). There is a total number of about 500,000 readout channels, giving a typical event size of about 220KB/event before compression. The typical sustained readout rate was about 5KHz for the Copper-Copper system, and 5KHz for the the proton-proton system.

The computers in the data acquisition are running the Linux operating system. They are connected through a Gigabit network, wit a 10G uplink to the HPSS-based tape robot system which is located at the RHIC Computing Facility about 1.5 miles away from the experimental site. The incoming data are first stored on local buffer buffer disks in the countinghouse. This allows to ride out short service interruptions of the tape robot, and allows to send the data to the tape system at a steady rate, leveling the ebb and flow of the data rate from the data acquisition, which tends to deliver data in bursts. The buffer disks have a capacity of about 40TB, which allows the buffering of about 36 hours worth of data.

During this time, the data are also available for various monitoring, calibration, and early-stage analysis tasks.

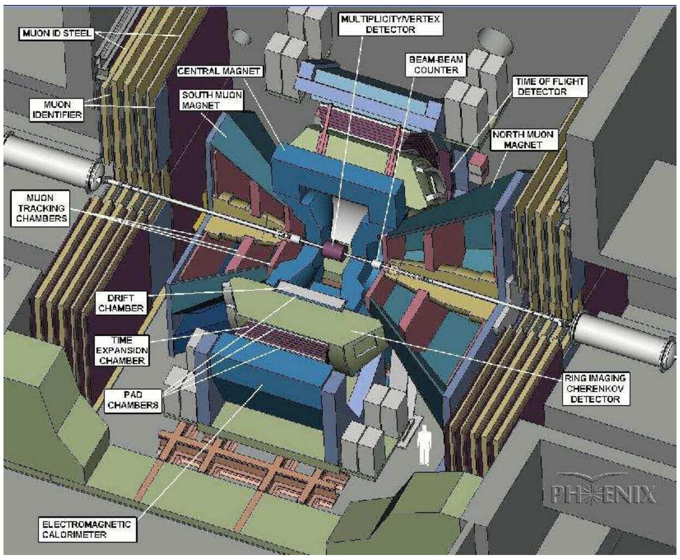


Figure 1. Overview of the PHENIX experiment.

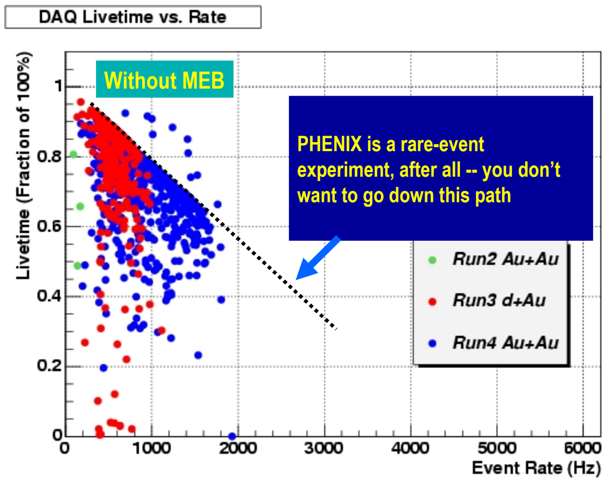
2. Ingredients To Achieve the Highest Performance

Over the previous 3 Runs, we introduced new features or removed bottlenecks and steadily improved the maximum event rate and data handling capability. We also improved the efficiency of the operations and increased the uptime of the experiment. Among the key improvements are

- Multi-Event Buffering
- distributed data compression
- improvements in the data handling code
- improved error recovery procedures

2.1. Multi-Event Buffering

One of the major accomplishments in Run 5 was the commissioning of the so-called *Multi-Event Buffering*. The front-end electronics stores the data samples in analog memory units (AMU), which hold the data for about 40 μs until a Level-1 trigger decision is formed. Once the trigger decision arrives, the digitization of the respective memory cell starts. The Multi-Event buffering allows to store up to 4 events in the AMU before raising the busy, which leads to a significant reduction in the data acquisition dead time. The effect is shown in figure 2. One can see the almost linear relationship of the maximum DAQ livetime as a function of the event rate for the Runs without the Multi-Event buffering. Since PHENIX is a rare-event experiment, after all, one cannot easily increase the data rate at the expense of DAQ livetime. In Run 5, with the multi-event buffering working, the data points are well to the right and above the line, showing a high data rate with a simultaneous high DAQ livetime.



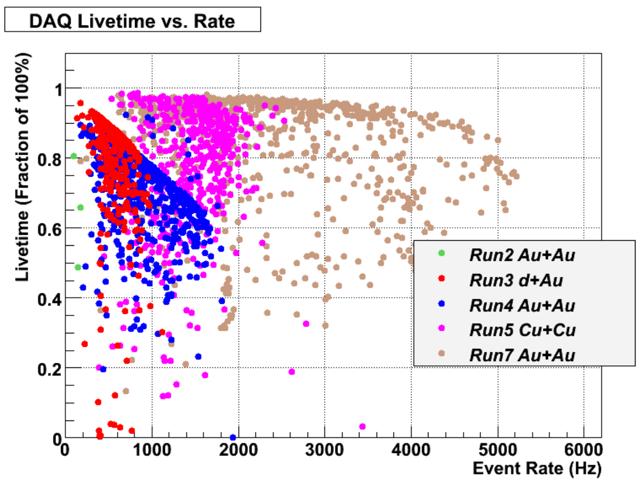


Figure 2. The effect of the Multi-Event Buffering. The figures show the data acquisition livetime as a function of the event rate for various runs. In the upper figure, for our runs without multi-event buffering, a sharp edge can be seen that represents the almost linear dependence of the maximum livetime for a given event rate. In the lower figure, multi-event buffering significantly increasing the livetime, the the points from Runs 6 and 7 extend well across that line.

2.2. Data Compression

When the PHENIX Raw Data Format (PRDF) was specified, we found that utilities such as *gzip* could further compress the already zero-suppressed data by more than a factor of two. Using gzip however, one would need to restore the whole data file to its uncompressed length before reading it, which is undesirable. We therefore specified a compressed raw data format where the data could be read and uncompressed on the fly. The raw data stream consists of buffers of about 16MB but variable length, and the length is always a multiple of 8KB. A typical buffer holds between 50 and 100 events. Originally this was put in place to increase the data integrity (if a corrupt buffer is encountered, one can salvage the subsequent buffers but skipping 8KB records until either the next buffer header or the end-of-file is found). The compression works on this buffer level. A complete buffer, including its header, is compressed by some algorithm.

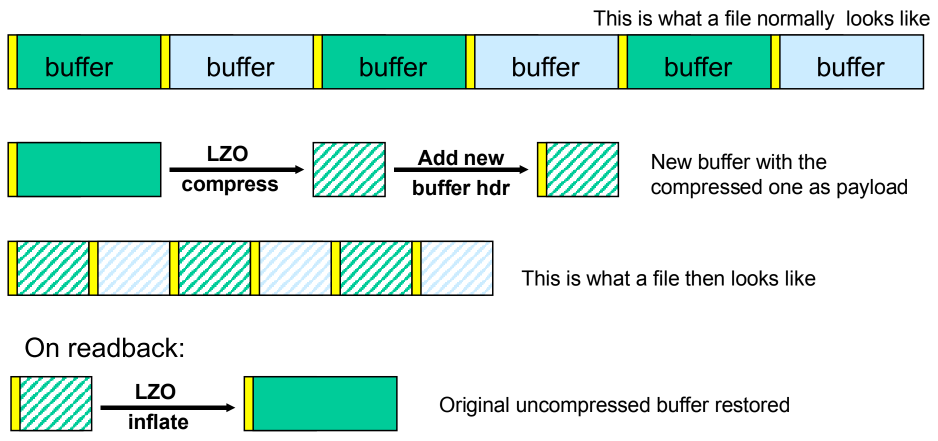

This is what a file normally looks like
buffer
buffer
buffer
buffer
buffer
buffer
LZO
compress
Add new
buffer hdr
New buffer with the compressed one as payload
This is what a file then looks like
On readback:
LZO
inflate
Original uncompressed buffer restored

Figure 3. The compression works on buffers. A complete buffer, including its header (denoted in yellow) is compressed. It receives a new buffer header, and is written out this way. On readback, the original buffer is restored.

The result, typically much smaller than the original buffer, gets a new buffer header and is written out to the file instead (fig. 3). The buffer header makes this a legitimate buffer as far as the transport layers of the system are concerned. On readback, the buffer is recognized as as one which holds another compressed one as its payload, which gets uncompressed. The original buffer is thus restored, and passed on to the next software layer, just as if it had been read from the file. In this way, the compression is handled at a relatively low layer in the I/O system, and completely transparent to the rest of the system.

Traditionally, we used the same algorithm internally used by the gzip utility, "compress2", which is readily available on most systems. However, while this algorithm achieves a good compression ratio (the compressed buffer shrinks to about 45% of its original size), it is also rather CPU-intensive, which makes it too slow to be usable in the online system. We searched for a different algorithm which has properties comparable to compress2, but is much faster. We wanted an open-source algorithm which is robust, adheres to a published standard, and is available on most systems. We identified the family of the *LZO* algorithms [2] as a good candidate, and settled on the *lzo_1x* algorithm. It achieves a compression of about 60% of the original size on a typical buffer (slightly worse than compress2) but is at least 4 times faster (to compress one minute worth of data, *lzo_1x* takes one minute, while *compress2* takes 4 minutes).

This is still a considerable amount of CPU cycles, but the work is distributed over all ATP's in the system, which compress each buffer before sending it to the logger.

In this way, we fit about 40budget. In addition to increasing the number of events, this has allowed to postpone the turn-on of the Level-2 trigger. Rather than discarding events, we have used the Level-2 trigger to write interesting events out to a different data stream, which is made available for priority analysis.

3. Accomplished Rates

The improvements are visible in figure 4, which shows the combined data rate in MB/s and a function of time, for the first 4 hours of a RHIC fill. This is the data rate that was written to the buffer disks. At the highest luminosities at the begin of RHIC, we are able to write close to 600MB/s of data to disk.

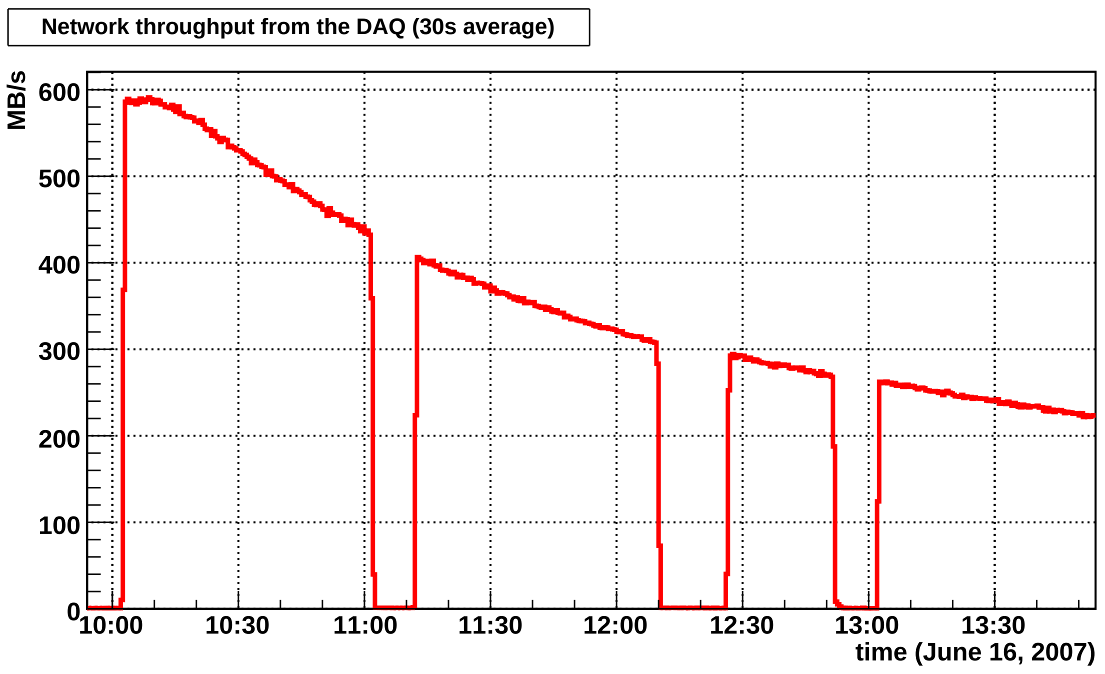


Figure 4. The aggregate data rate, in MB/s, that was written to disk during a RHIC fill. At the highest luminosities at the start of a fill, the data rates is almost 600MB/s. The gaps in data writing are the starts of a new run of the data acquisition.

References

[1] K. Adcox et al, *PHENIX detector overview*, Nucl. Instr. Meth A 499, 2003, pp 469-479.

[2] M. Oberhumer, *The Lempel-Ziv-Oberhumer data compression library*, http://www.oberhumer.com/opensource/lzo