The Fermilab DART Data Acquisition System at Running Experiments*

Gene Oleynik, for the DART collaboration: [a]
Online Systems Dept., Electronics Systems Engineering Dept., E781, E815, E831, KTeV, E835, E871, E872

[a] *Fermilab, MS 369, P.O. Box 500, Batavia, Il., 60150*

The full DART system has been running since June, when the first beam was delivered to the Fixed Target program. We report on the completion of the DART system, experiences at the experiments, and plans for the future. We concentrate on those portions of DART that were completed towards the end of the development, in particular our graphical monitoring system, damp, which is of general use.
The DART data acquisition system is designed to meet the needs of a number of experiments with a wide range of requirements: from a few hundred KB/s at low rates to one of the highest rate HEP data acquisition systems in operation - KTeV. What stands out in DART is its fully distributed nature, its high degree of tailorability, and its wide range of readout rates and throughput, while at the same time maintaining high levels of integration. We report on status of DART at the experiments.
One of the more recently completed and enthusiastically received pieces of DART is the damp DA monitoring framework. This is a system that allows distributed applications to provide dynamic statistics for DISPLAY through a simple API. The data is collected and displayed through a tcl/tk graphical program. This program can display the data as multivariable strip charts, bar charts, or numeric scalers. A damp interface builder lets one construct customized displays and their associated data graphically in a matter of minutes.
We also review experiences gained from use at running experiments and extensions to DART that are under development for the future.

Key words: Data Acquisition; Monitoring; Fermilab Fixed Target Program

* This work is sponsored by DOE contract No. DE-AC02-76CH03000

1 Introduction to DART

DART [1,2] is a collaboration between Fermilab experiments and the Fermilab Computing Division to provide a generally usable DA system for the current Fixed Target runing period. The run started ramping up in June, and eight experiments are currently taking data with DART. The experiment data acquisition systems have performed reliably since the start of commissioning. The strategy of incremental versions of DART to support commissioning and then full data taking has proven to be successful and resulted in data acquisition not becoming a critical path item for any of the experiments. As of February, the high rate KTeV experiments has recorded approximately 2.5 billion events onto 20 TB of dlt tapes.

In DART, front-end readout controllers push data over multiple RS-485 “stream” readout cables into programmable dual ported buffer memories residing in VME “event building” crates. A single event spans across streams, and its pieces are directed to a single VME crate. Events are cycled through VME crates in a round-robin fashion, and each VME crate is connected to a SGI Challenge processor via a Performance Technology (PTI) high speed VME-VME link. The Challenge reads the event table and fragments from the buffer memories into its own memory, which is managed by DART software.

The highest bandwidth user is KTeV, which has 6 streams, 3 event building VME crates, and an online analysis crate, for a total of 24 buffer memory nodes.

DART is fully distributed; it was designed to start up, control, and monitor an arbitrary number of computer nodes, and to handle replicated nodes efficiently. It is also highly customizable at all levels. DART provides a mechanism to define and automatically start up multiple DA configurations. Application control is performed in a highly parallel, but synchronized fashion. DART software also provides a distributed information system for configuring the DA and recording a run history; a system for local buffer management and providing services on the buffers; a data logger for tape and disk files; an error reporting, logging, and display system; a system for providing back-end workstations with live data over the Internet and an analysis framework to process the data and display it in PAW or HISTOSCOPE; and lastly, a graphical monitoring system for monitoring the entire DA.

2 The DA monitoring Program – DAMP

Damp is the DA monitoring system developed for DART. It provides a means for distributed DA applications to update displays with performance and other monitoring statistics, and a flexible, easy-to-use means of constructing displays and associating statistics with them. Applications make calls to register monitoring parameters which are identified by strings. They are returned local shared memory locations to store the data associated with the parameters. The parameter names for each application are only sent to the server once so that only data is sent to the server on updates. A process that sits on each node, called the forwarder, periodically reads the data out of shared memory and ships it to the server. As a consequence, the DA applications are decoupled from server or forwarder crashes.

The damp server is just another instance of our Distributed Information Services (dis) server, which is also used for providing DA configuration information as well as recording a run history. The basic functionality of dis servers are to store and fetch key(string)-value pairs from a "database". There is one significant difference for the damp instance: for fast access, the database is in memory instead of on disk (though it can be run from disk).

damp displays connect to the damp server and periodically fetch parameters for updating displays. Currently, the damp display provided by DART fetches all parameters (using a dis wildcarded fetch call for network efficiency).

The damp display uses the public domain tcl/tk system and blt tk widgets, in addition to custom widgets built upon these. The types of widgets supported are multi-value strip charts, multi-value bar charts, and scalers. Strip charts can be populated with time-stamped arrays to get time plots.

We also use the public domain tk graphical interface builder, "gui builder" [3]. We have added to its capabilities menus to associate damp parameters with display components such as strip chart lines and bar chart bars. This program makes it very simple to graphically add a display widget from a palette to the display, associate damp statistic(s) with it, and save the configuration.

We provide experiments with a damp display template (for displaying command "standard" damp statistics), the damp server (started up automatically by the DA on the host node), and canned applications such as the logger make calls to store standard logger damp statistics. The experiments provide damp initialization and store calls in their applications for the DA statistics produced by the application, a call to invoke the damp forwarder on the nodes which will produce damp statistics, a call to start up the damp display on the host in the host script, and customization of the display for their DA using the gui builder.

Damp has been enthusiastically received by the experiments. Each has customized their damp displays such that on first glance, one would not imagine they were constructed from the same system. One experiment, E872, is using damp to display slow control information as well as for DA monitoring.

3 Status and Experience

DART experiments have been taking data since June. Overall, DART has been extremely robust. We get a couple of support call per week now, and these are for the most part computer hardware failures.

The DART hardware has been quite stable. A few ECOs had to be made for adjusting to the realities of experiment environments (e.g. ground loop and transmission line effects). At one experiment fiber optic transmitter ICs were operating below their design temperature due to the Chicago winter air coming through a cooling vent. In additon, we have had to resolve the usual problems with additional or missing data words.

For DART software, we had numerous low level memory leaks or segmentation faults when the run first started. These ranged from once every few hours to once per month. because of their low frequency and non-local nature, these bugs are very difficult to track down. A couple of months into the run, we started running our code through a commercial tool called Insure++ [4], which detects leaks and memory overwrite problems. We ran it on older versions of our code for which we had already fixed a number of problems - it found all of those plus more. The pinnacle for this tool was solving a once per day crash in damp that was occurring at only one experiment. It took a weekend of running the experiment damp configuration on our test-stand before the code path was traversed and the bug was identified by the tool as a double memory free. Since then we have "insure-ified" all of our core software. If there are still such problems, their frequency is very low. We recommend this tool for C and C++ software, be it online or offline.

An unexpected difficulty occurred with our Operator Control Program (OCP), which was designed in tcl/tk to be highly customizable. This turned out to be a two edged sword in that local experiment customizations resulted in conflicts when we made global additions or corrections to the product and wished to propagate them to the experiments. The procedure we have adopted is to analyze their local customized startup file and manually add changes to it. This is less than satisfactory. Reorganizing the startup files into "user" and "internal" portions is not fully satisfactory either. Where we did do this, an experiment customized the internal as well.

4 Current Work and Future Plans

The currently supported links from the "event building" VME crate are the PTI VME to VME, or Ethernet. While the slower ($<$ 400 KB/s) Ethernet option is free, greater performance currently requires the DA computer to have a VME bus, such as an SGI challenge, and this is expensive. We are now exploring using the BIT3 link, which supports other busses such as gio, to fill this performance gap. This will permit cheaper filter/logging computers, such as an SGI Indy, to be used. Since the BIT3 does not support VME D64, rates of up to only 20MB/s are available.

We plan to add a graphical editor for editing system configuration parameters.

There is currently no logger state display except for the statistics fed to damp. We are working on a logger state module to add to the operator control panel.

We plan to port at least part of the software to NT and/or Linux.

We are currently working on integrating DART into a number of test-beam DAs. Integration with EPICs in the D0 test-beam is ongoing, and integration with older parts of the CDF Run I DA went quite well.

5 Conclusion

DART has been successfully incorporated into eight experiments which comprise a very wide range of requirements, rates, and architectures. Our goal to provide a general, highly customizable and robust DA system to meet the needs of these experiments has been achieved. Except for a few teething and a few steady state but low frequency problems, and the addition of a enhancements, the run has so far been going quite well.

6 References

[1] "Extending DART to Meet the Data Acquisition Needs of Future Experiments at Fermilab", Proceedings of CHEP95

[2] www-dart.fnal.gov:8000 is the DART collaboration home page.

[3] Graphical User Interface Builder, Sean Halliday

[4] Insure++ is a product of ParaSoft Inc. www.parasoft.com