Gamma-ray Imaging Framework Overview

Austin Benson

Originally written January 30, 2011
Updated: March 17, 2012

1 Gamma-ray Imaging Framework (GRIF)

1.1 Introduction

The Gamma-ray Imaging Framework (GRIF) is a software project for the Berkeley Applied Research on the Imaging of Neutrons and Gamma-rays group (Bearing), which is a subset of the Domestic Nuclear Threat Security Initiative (DoNuTS). The software is designed to be parallel and multi-platform. The application framework is comprised of data acquisition threads, "DAQThreads", and data analysis threads, "AnalysisThreads". Small applications are expected to run around 10 DAQThreads and 5 AnalysisThreads concurrently. There are three main projects for which the framework will be used: computer vision for nuclear detection, the High Efficiency Multi-mode Imager (HEMI project) [1] at LBL and SSL, and Gamma-ray detection with Sodium Iodide detectors.

The problem with serial nuclear detection software is that serial acquisition and analysis of data is extremely slow. As detection becomes more advanced, the scale of projects increases immensely. Projects often scale in parallel with multiple copies of detectors. Thus, larger projects are suited to parallel software. The nuclear detection community must adopt largely parallel systems in order to push projects forward.

1.2 Motivation

The primary goal of the GRIF project, and for the DoNuTS group in general, is to improve techniques for nuclear detection. Efficient nuclear detection provides an international security tool. Nuclear detection systems can be set up in transportation areas such as airports and train stations in order to prevent the distribution of illicit nuclear material. However, efficient systems require parallelism, so new nuclear detection techniques will require an interdisciplinary effort by computer scientists, physicists, and nuclear engineers to develop parallel software.

1.3 Performance and Simulation

The GRIF software is currently in alpha testing, so large-scale project performance data is not yet available. However, there has been extensive simulation conducted to understand performance. The benchmark for analyzing performance of the system is a multiple-channel simulation of energy peaks and background noise. Each channel detects different energy peaks with different levels of background noise. This simulation uses two DAQThreads and one AnalysisThread running concurrently. The DAQThreads produce data, and the AnalysisThread produces and updates histograms in real-time. Figure 1 shows the visual display of the simulation. Right now, the simulation runs smoothly with 300-1500 data packets posted to memory every second.

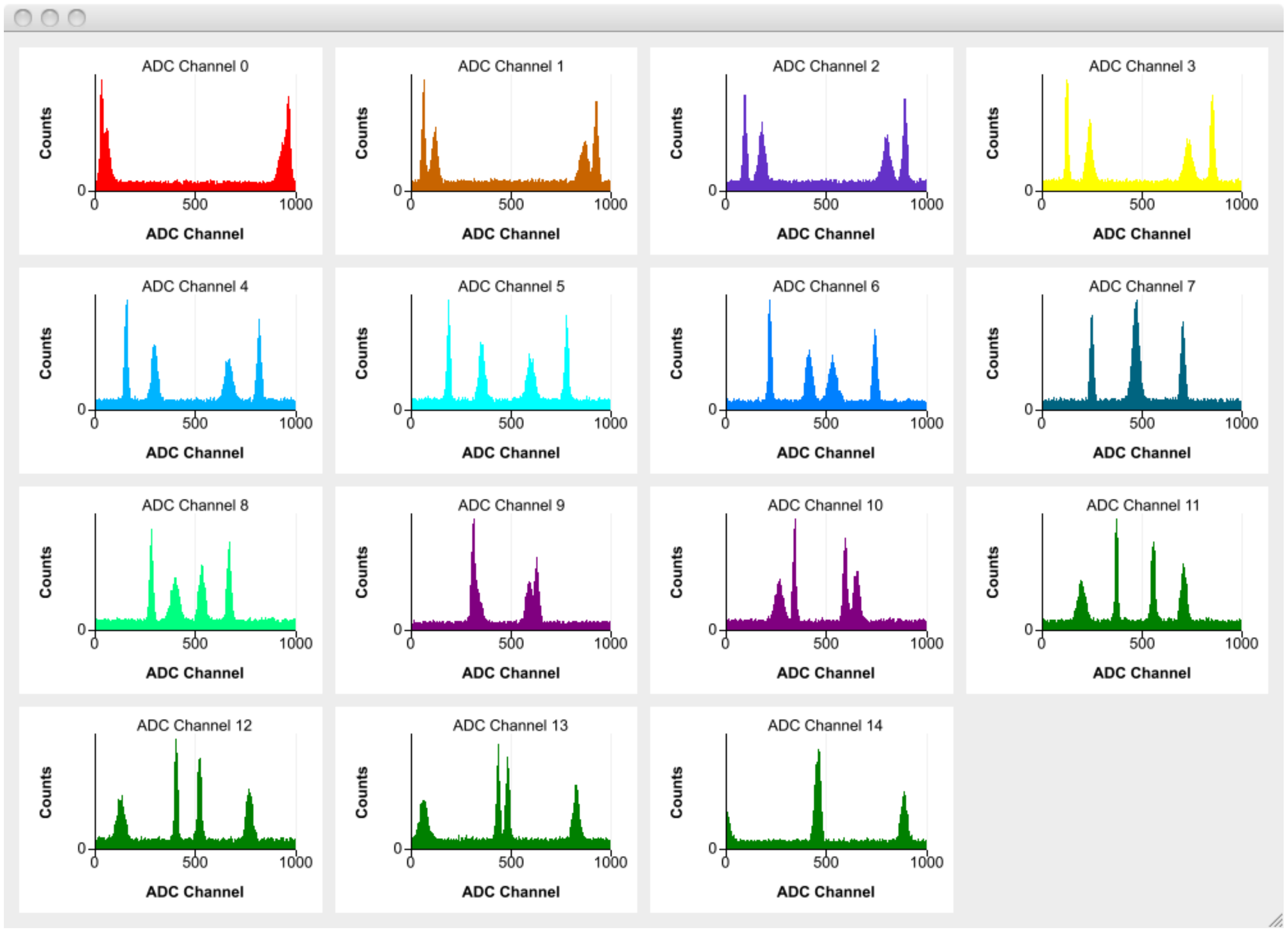

Figure 1: Simulated data of 15 channels (histograms are updated in real time)

1.4 Challenges

The most significant challenge of the project is memory management with load balancing. Figure 2 diagrams the slice of memory for one DAQThread and one AnalysisThread running concurrently. The block is the type of data being pipeline, e.g. energy values. The buffers accumulate data packets, which are obtained from hardware (in applications) or software (in simulation). Load balancing between threads and data buffers has been a difficult challenge. We currently run a priority scheduler, where DAQThreads and AnalysisThreads that push more data through the system are given a higher priority.

1.5 Tools

GRIF is C++ software developed on Qt [3], an open-source user interface framework maintained by Nokia. We use Qt because of its graphical user interface and platform-independent multi-threading tools. Qt's QThread class provides a platform-independent thread abstraction. Thus, we can compile multi-threaded projects on Windows, OS X, and Linux distributions. QThread has been an invaluable resource for the multi-platform project. Qt also provides its own mutex, semaphore, and other concurrency-managing objects in a platform-independent setting. In addition to Qt, we use the scientific libraries, we use the Boost Graph

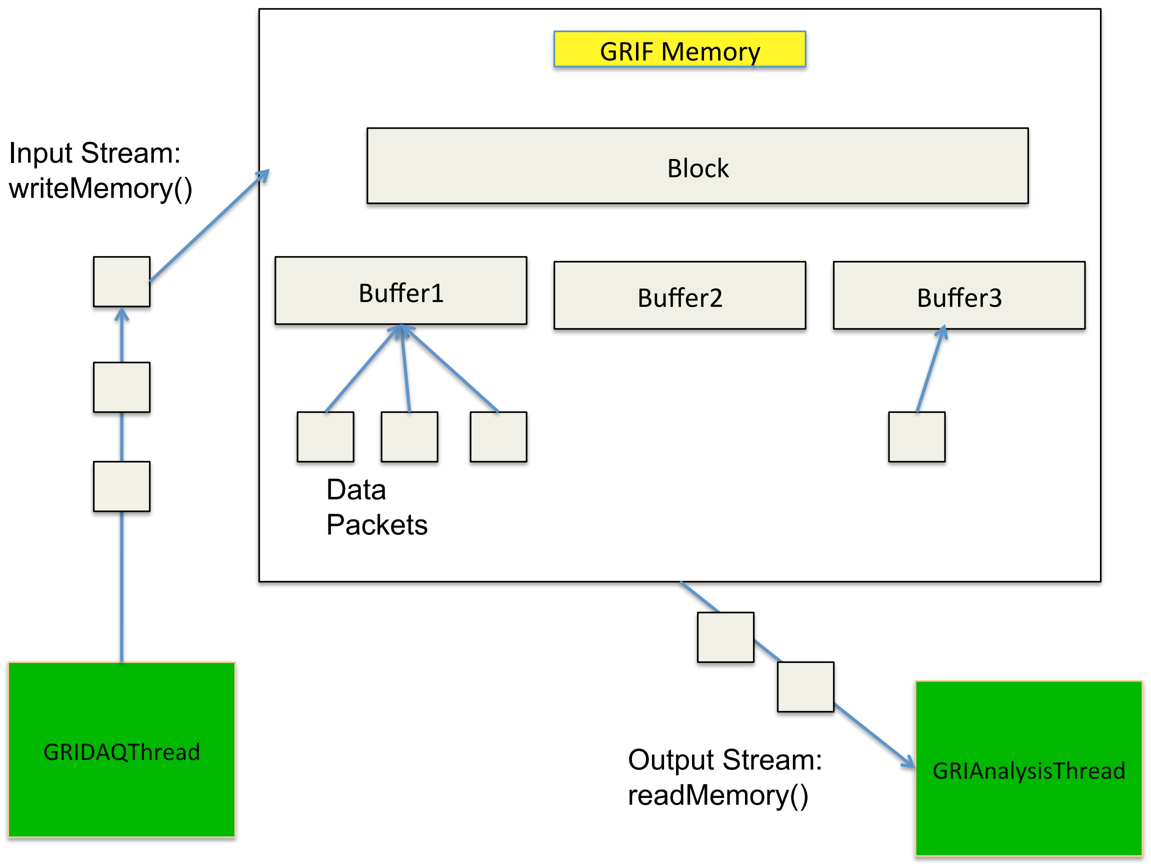

Figure 2: GRIF Memory System

Library (BGL) [2] and the scientific library ROOT [4].

1.6 Scaling

One of the long-term goals of GRIF is to use the software on large distributed systems. For example, one idea is to put nuclear detectors on New York City taxis and analyze the data in real time to detect nuclear threats. A simpler project is to have a smaller, local distributed system of around 10 machines (each with a quad-core processor) to analyze nuclear detection data in a laboratory setting. The idea would be to have a remote machine as the system master with a cluster of smaller computers performing computation. The master computer would have a graphical user interface to monitor the data on the cluster.

The scaling of GRIF to these projects has not been accomplished yet because of complexity and resource time. The software is developed mostly by undergraduates in EECS, so developing large systems would take a large amount of planning and time. Another challenge with large scale problems is that the amount of data rapidly increases. The massive data must be handled by more threads, more machines, and more parallel software. This requires more load balancing and careful memory management.

References

[1] A. Zoglauer. *The Nuclear Compton Telescope NCT and The High Efficiency Multi-mode Imager HEMI.*

⟨ NCT and HEMI ⟩

[2] Boost Graph Library
⟨ www.boost.org/libs/graph ⟩

[3] Qt: A Cross-platform application and UI Framework
⟨ http://qt.nokia.com/ ⟩

[4] ROOT: A data analysis framework
⟨ http://root.cern.ch/ ⟩