

Future of DAQ Frameworks and Approaches, and Their Evolution towards the "Internet of Things"

Niko Neufeld

PH Department, CERN, Geneva, Switzerland

E-mail: niko.neufeld@cern.ch

Abstract. Nowadays, a DAQ system is a complex network of processors, sensors and many other active devices. Historically, providing a framework for DAQ has been a very important role of host institutes of experiments. Reviewing evolution of such DAQ frameworks is a very interesting subject of the conference. "Internet of Things" is a recent buzz word but a DAQ framework could be a good example of IoT.

1. Introduction

Data Acquisition frameworks are very susceptible to technological changes. Since data acquisition is so close to the hardware, it has to adapt constantly to the dramatic technological evolution which we are witnessing everyday. In this respect data acquisition frameworks are similar to operating systems. This is probably one of the reasons why these frameworks have often a comparatively short life-time. With the exception of Unix - and to a lesser extent Microsoft Windows - all operating systems which ever had a certain hold in the market have vanished again, or have been banished into niches. DAQ frameworks as well are reinvented ever so often. In this paper I would like to review a bit the status of DAQ systems in 2015, I will try a look ahead in the near future, where I think it is safe to venture some guesses, and I will offer some speculations of how certain upcoming trends and technologies, such as the Internet of Things (IoT) might influence large-scale data acquisition systems.

2. DAQ frameworks in general

In this paper DAQ framework will be used almost synonymously to DAQ system, because the shift to more and more commodity based hardware, means also that the emphasis in DAQ is more and more on the firm- and software and configuration of standard components, which in turn means that "framework" is a very appropriate generic term to talk about DAQ. Functionality typically expected from a DAQ framework includes:

- (i) Data transport from the detector
- (ii) Event-building in case of multiple data-sources
- (iii) a job control for the typically many software processes involved
- (iv) Some interface for monitoring of / spying on inflight data for the purposes of quality control and debugging



- (v) A control-system to steer the acquisition. It also should interface to and/or synchronise with important external systems
- (vi) An interface to data processing
- (vii) An interface to long-term storage

DAQ frameworks normally provide all or at least a large number of these functionalities. It should be noted here that in principle SCADA systems (Supervisory Controls And Data Acquisition) such as EPICS, Wincc-OA and many others, which are popular in the controls-world offer much of the same. The difference is mostly in the emphasis on certain functionalities. In data acquisition the most important task is the movement of the data, and the amount and/or rate of data is what determines the technical complexity, size and cost of the system. In SCADA systems the emphasis is more on time-ordered recording and behavioural modelling. When there is a modest amount of data and a SCADA is already in place, then it would seem logical to use it for the DAQ as well.

3. Size matters

One can attempt to broadly classify DAQ frameworks into three classes depending on the size of the system. The first class comprises systems which can be essentially readout with a single system (for simplicity one can think of a single PC-server). If all links from the detector can be collected with a single data-sink, many of the features of a DAQ system can be simply done using standard OS tools. They can be “glued together” using scripting languages such as shell-script or VisualBasic, or one can use any of a host of commercial offerings such as NI’s LabView. Obviously also the DAQ frameworks discussed in the next section are perfectly usable in such a small-scale context.

It should be noted that such a system is not necessarily “small” by historical standards. A high-end modern PC can achieve more than 400 Gb/s of bi-directional I/O. But one does not even need to go so far. Already 40 Gb/s I/O card corresponds to 100 kHz of 32-bit samples of 10000 channels! This is significantly more than for example all LEP experiments combined [1]¹. Such an argument presupposes that the samples are aggregated somewhere before they hit the main CPU(s)².

Such a single-system setup will be able to store about 1 GB/s on traditional spinning drives³ or much more on solid-state drives (SSD).

The amount of processing which can be achieved depends obviously on the algorithms, but with multiple cores at high clock frequencies simple operations can be done even on very high rate data-streams.

Such a simple yet powerful DAQ, could be setup in principle in a single application which reads data through the OS, does some processing and writes them through the OS to a file. Steering can be added easily in some asynchronous loop.

As soon as the resources on a simple system do not fit the complexity of the setup increases significantly In the *second class* multiple data-sinks require the orchestration of data-movement and bookkeeping, a process usually referred to as *event-building*. Start-up and tear-down of processes is now in several places and in some cases synchronisation between these processes is required.

¹ One can say that “Moore’s law” in the sense of “exponential growth of information technology resources” holds here too.

² The message rates and consequently the interrupt rate, must be kept low, because high interrupt rates will inevitably lead to very high, inefficient resource usage on a CPU. High-performance network interface cards (NIC)s and the Linux and Windows network stacks devote a lot of ingenuity to mitigating interrupt load.

³ This assumes eight drives in a RAID configuration with about 150 MB/s per drive.

At this point the software development effort for setting up a DAQ will require a substantial effort. Still as long as the system is not too large or requires exploiting specific features of a specific technology, a lot of effort can be saved by using one of the available generic frameworks.

3.1. Generic DAQ frameworks

Generic DAQ frameworks are tool-kits to build DAQ systems of practically any size. They offer technical solutions for all the tasks of a DAQ and provide the necessary “glue”. They also come usually with guide-lines, configuration and build tools. In general they will allow to reduce the development effort spent on technical details. The ubiquity of certain key technologies is an important factor, which makes such frameworks “universal”. These technologies are

- The x86-based PC
- Ethernet and TCP/IP based networking
- The Linux operating system

Let us briefly review why these technologies are so important. All of them are very old technologies: the x86 PC is more than 30 years old, Ethernet and TCP/IP almost 40 and Linux is the last and most successful offspring in a family of OSes which started more than 40 years ago. Hundreds of thousands of engineers have known, developed and worked with and for these. The literature is boundless. All of them are open: every detail is known, in principle you could re-create all of them from the existing documentation, even though this would not be an easy undertaking. All of them are also to some extent community owned standards: the PC revolution only really started once the BIOS was re-implemented and so no single company really controls the x86 PC, even though it sometimes seems as if a very powerful player would like to take full ownership long after IBM again.

All of them are “industry-standards”. I would like to contrast this with “committee” standards. Committee standards are created usually by bodies, whose members have an interest but sometimes no vital stake in the technology. They tend to be bloated, many-facetted and never to be properly implemented. The OSI model is such an example, often quoted, in practice not very useful. International bodies such as the ITU and ISO are a fruitful source of such standards. In the best cases they bless something which the “industry”, i.e. the people in practice actively working with a technology have come up already, e.g. the ISO C standard. One could think that Ethernet is an example to the contrary, but Ethernet is not really owned by the IEEE, which in itself is just a forum for industry players to settle on common standards. Where the IEEE moves too slowly, it can be seen that industry takes its own steps very quickly: examples are the upcoming 25 and 50 Gigabit Ethernet standards, which are driven by the availability of serialisers of that speed and 5 Gigabit Ethernet which is driven by pragmatic cabling considerations in data-centres.

Yet despite the informal, “collective” ownership all these standards have a kind of “core” which is well-defined and makes them well-defined enough to be universally useful. A PC is a PC, a well-defined platform, and has been so since decades. You can boot DOS or Linux and the core platform will work, even on the latest Haswell processor. This is no little advantage: part of the “start-up” problems of non-x86 architectures is this lack of being a PC. Obviously one can get an ARM or Power9 server to boot, but today at least each platform requires special firmware and has its own (small) eco-system, necessitating a fair amount of repetitive setup and integration work.

3.2. Description of a few generic modern DAQ frameworks

A very brief overview of four DAQ frameworks should illustrate the preceding discussion. More information can be found on the home-pages of the projects.

3.2.1. MIDAS MIDAS [2] developed at the Paul Scherrer Institute, now maintained by TRIUMPH, is more than 20 years old. It is very portable running on Linux, MacOS and Windows and is in use by a large number of experiments. It uses TCP/IP for data-aggregation and is integrated with the popular ROOT package for analysis and data treatment. It also includes a run and job-control and state-machine tools to build a complete DAQ system.

3.2.2. DAQ middle-ware DAQ-middleware [3] is a distributed C++ framework for DAQ developed at KEK. It is available for many UNIX flavours and like MIDAS uses TCP/IP for aggregation. It uses web-technologies for user-interfaces.

3.2.3. ArtDAQ artdaq [4] is a modern DAQ framework developed at FNAL for experiments at the intensity frontier. It builds on the ART physics software framework developed for the same community. artdaq uses MPI for data-transport, and can thus seamlessly work on both Ethernet and also InfiniBand, which is useful when very high data-rates are required. It is a complete toolkit, with process-control, state-machine, etc... which experiments use to build a complete DAQ.

3.2.4. !Chaos !CHAOS [5] is different from the other described systems, because it comes from the controls world. It uses a noSQL data-base as a backing-store for the data and a distributed RAM-cache as a buffer. Data go over TCP/IP in BSON format.

!CHAOS is mentioned here also because SCADA systems can be very large DAQ systems as demonstrated by the NIF and ITER control and data acquisition systems. For NIF the system consists of more than 500 servers monitoring many tens of thousands of devices [6]. At ITER the DAQ will be capable of a peak-performance of 50 GB/s, which is similar to the current generation of LHC experiments. Archival rates can reach up to 1 PB/day [7, 8].

3.3. Large experiment frameworks

Large experiments often have their own frameworks. Apart from sociological reasons - these experiments have often a small team of full-time professionals responsible for the data acquisition - the main two reasons for a dedicated framework seem to be:

- (i) A very tight integration with the overall experiment software
- (ii) The wish to exploit to the maximum the properties of a specific technology

An example for the first case is the current DAQ framework of the LHCb experiment [9], where the software trigger was designed from the beginning to use the exact same software, GAUDI [10], as used for the reconstruction and analysis. As an extreme example in the current LHCb DAQ the event-builder software is simply a specialised “algorithm” in a standard generic framework process. All build-tools, compiler versions and software interfaces are the same, with a few adaptations for use of facilities outside the traditional batch-processing. It would have been difficult to fit an existing framework, which also “operates the main-loop” into this.

An example for the second case is the data acquisition framework for upgraded LHCb data acquisition for the LHC Run3 and beyond [11]. The goal here is to implement a cost-efficient event-builder capable of building 40 Tb/s. In the base-line architecture there will be 500 event-builder nodes each of which will see an aggregated I/O of about 400 Gb/s. While this is possible with modern Intel-based servers, it requires absolutely zero-copy operations, alignment of data-accesses, cache-friendly auxiliary structures and a very tight integration with the FPGA code of the acquisition card. No generic framework can be reasonably expected to support such a use-case out of the box and extending a framework so far is most likely as much, if not more, work as writing a dedicated one. There is of course a cost involved in writing a DAQ framework,

however in the author's experience most of the time to get data-transport and event-building software running both correctly and efficiently does not go so much in the actual code, which is after all algorithmically rather simple, but in the working around all kind of idiosyncracies of the hardware and various protocols.

3.4. Trends in large DAQ systems

The largest DAQ frameworks are currently the LHC DAQs. They will be "joined" by the DAQ frameworks of the FAIR experiments around the time of LHC Run2. The DAQ for Belle-2 is somewhat in between and characterised by one very large contributor requiring special treatment (the Pixel detector PXD)⁴.

All these experiments are characterised by the enormous amount of data, both produced by the experiment and in need of online filtering and of permanent storage.

3.4.1. Reduction of "raw" data One way to tackle the long-term storage problem is to not store all the data originating from a collision or a time-slice. This amounts to "throwing away" or "not keeping" the "raw" data, and is therefore a very sensitive topic. It should be kept in mind of course that the "raw" data are always a construction - decisions in the front-end processing will determine what will be the raw-data stored on tape. Examples include cluster-finding in calorimeters and tracking detectors.

Looking at what is actually used in analyses one can of course think of going one step further and store even more "derived", higher-level quantities. In order not to jeopardise the physics output of the experiment, this requires that the same or nearly same precision can be achieved in the online processing as it can be in offline processing using lower-level, more primitive data.

This will require calibrations and alignments as good as the ones available for offline processing. Naturally in such a scheme the offline algorithms, or maybe slightly adapted versions, which are equivalent in physics performance, will be run in an online context. This leads to a very close intertwinement of on- and off-line processing and code.

The DAQ frameworks must then provide means for fast offline quality calibrations and alignments. The necessary co-design for on- and off-line code and frameworks are most stringently realised in the upcoming ALICE DAQ framework for Run3 and beyond, called O2 [12].

3.4.2. 24/7 usage of Online facilities The Online computing infrastructures of the LHC experiments represent a substantial amount of computing power, larger than most Tier1-sites and also at least as large as HPC systems in the upper 25%. If one wants to stick to the Grid terminology, one can, at least for the case of the CERN based experiments, argue that they are really Tier-1s since they can easily and transparently access the tape-facilities provided by CERN so they fulfill then all the requirements including a mass-storage infrastructure.

Since the uptime of no accelerator is 100%, and for the LHC it is between 25% and 35% averaged over a year, it would be obviously wasteful to use the facilities only for online processing. All LHC experiments have been starting to use this extra capacity already during Run1 of the LHC, for Run2 this will be continued and improved and for Run3 this has become a major design criterion for both the ALICE and LHCb upgrades.

One can discern two uses of the Online facilities outside synchronous online processing:

- (i) offline processing
- (ii) asynchronous online processing

⁴ Of course domination by a single subsystem is not unusual: e.g. the ALICE TPC is accounting for almost 90% of the data and similarly the ATLAS calorimeter is the main driver of the event-size there.

Offline processing is a rather simple use of online resources. Here the compute and storage resources are presented either as a private cloud (ATLAS, CMS) or as a simple batch-farm (LHCb).

The coexistence with real online processes is made difficult by the typically long running time of “offline” jobs, oriented towards batch-farms. These jobs, at least today, cannot be simply suspended and resumed many hours later, although in principle nothing speaks against this. In practice this means it is difficult to use shorter no-beam periods and that the queues need to be drained well in advance of a restart of data-taking activity.

Since the software basis is largely the same in the Online and Offline world it is possible to run on the bare-metal as in the case of LHCb. Alternatively one can run on virtual machines (ATLAS, CMS) with the advantage of better isolation, at the expense of slightly larger overheads.

Asynchronous online processing at the LHC was pioneered by LHCb. Asynchronous means that the final event reconstruction and selection is done a significant and “random” time after the event has actually been recorded. The delay until final processing will be determined by the availability of computing resources and/or the availability of certain meta-data, like for instances calibrations. In all these cases online processing continues beyond the actual data-taking, leading to a better use of the available compute resources. ALICE will make heavy use of this in its combined O2 system, planned for Run3.

For the GPD this can be a way to cope with the luminosity life-time, by “borrowing” CPU time early in the fill for later periods, when not all CPU is required. If the detectors can stand the luminosity this can be an alternative or supplement to luminosity levelling.

4. The future

4.1. Trigger-less read-out

With increasing complexity of events it can happen that traditional high p_t triggering becomes rather inefficient. Several large experiments plan therefore to operate trigger-free in the future. This includes CBM, which will have up to 10 MHz of interactions, 1 TB/s of data. The read-out will be continuous and self-triggered. FGPA pre-processed data, will be processed in time-slices and resolved by full online reconstruction [13].

Would this be an option for ATLAS and CMS? The main challenge for the GPDs lies in getting the data out of the inner tracking detectors at really high rates. Due to the radiation levels it is not possible to do this with high-bandwidth, low-power optical links but requires traditional copper cables. As has been nicely discussed by W. Smith [14] this would lead to a solid copper cylinder in the central part of the experiments and also cause dramatic cooling and power problems.

The only way out of this would be a high-speed, low-power, *very radiation-hard* optical or (highly directional) wireless link, which can be integrated in the readout-ASICs. Maybe some future form of SiliconPhotonics or something developed for the IoT⁵ can offer a solution to this problem.

In the absence of any realistic way to meet these challenges the GPDs have decided on improving the first-level hardware-triggers and augment them with a limited amount of tracking information (“track-trigger”).

4.2. Trends for the future

Looking at the proposals and plans for future large DAQs the following trends can be seen:

- a strong convergence to a PC-based, single-stage event-builder using Ethernet and/or alternative high-speed network interconnects;
- network bandwidth has become very affordable and this is being exploited;

⁵ In case it is *accidentally* rad-hard.

- standard protocols developed outside HEP are being used also for DAQ itself (0MQ, MPI);
- heterogeneous computing (FPGAs, GPGPUs, PHIs) (in particular ALICE, CBM) is actively studied and in some places prominently adopted;
- tight integration / fusion with “offline” frameworks; (LHCb, CMS for example and also ATLAS as of Run 2.)
- higher and higher trigger-rates or no trigger at all;

These trends will continue, driven by the technology (r)evolutions in the IT-industry and the needs for higher event-rates to look for extremely rare physics.

5. Technologies driving DAQ evolution

I will not discuss PC/server technology here, whose development is obviously crucial. Similarly I will skip the impact of accelerators (GPGPUs, FPGAs), because their impact is not specific to data acquisition.

5.1. Link-speed and LAN technologies

The speed of a single optical link has been steadily increased from 10 Gigabit to 25 Gigabit today. Often four such lanes are grouped to make one high-speed link (40 Gigabit and 100 Gigabit Ethernet are the most prominent examples). Recently 56 Gb/s has been demonstrated. For typical LAN applications the power for each such lane is less than one Watt. The first 200 Gb/s interfaces can therefore be expected in 2018. Somewhat surprisingly for short distances direct copper cable assemblies still work and are more cost-effective than optical solutions. This is one more reason to design very dense, tightly packed DAQ processing systems.

5.2. The Internet of Things (IoT)

Let us start with a definition:

”The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. IoT has evolved from the convergence of wireless technologies, micro-electromechanical systems (MEMS) and the Internet.”⁶

IoT combines embedded system with internet connectivity. It has become possible by extremely low-power microprocessors and a plethora of low-power wireless network technologies. Key characteristics are:

- identifiability on the Internet, requirning IP (TCP/IP) capable device with unique IP address;
- local intelligence requiring “micro-controller” of some sort;
- sensors and actuators implying normally a two-way communication;

IoT is one of the drivers of ”Big Data”, because of the ubiquity of sensors and the belief of many that a lot of value lies in the cumulate sensor data. The large amount of individual sensors and the huge amount of data are something which IoT has in common at first sight with large data acquisition systems. There are however also significant differences:

- Data Acquisition systems are traditionally hierarchically organized, and there are good reasons for this. The model for the IoT is the mesh, which does not mean that there are not also aggregation points of data.
- Detector channels are in a static configuration and identified by their location. Many IoT devices are movable with no fixed location.
- Detector channels are normally only sensors, not also actuators.

⁶ <http://whatis.techtarget.com/definition/Internet-of-Things>

- Normally there is no cross-communication between detector-channels, the data-flow is hierarchical.

There are other reasons why the IoT is a bit doubtful as a model for the front-end electronics of a many-channel, high-rate detector:

- The IoT addressing overhead is significant, because it uses normally an IPv6 header (40 bytes) + 20 bytes for the TCP header + some link-layer headers. Efficient link-usage under with large headers requires a high payload, this means that a lot of multiplexing needs to be done at the readout channel.
- Off-the-shelf IoT capability in micro-controllers or FPGAs is very likely to use TCP/IP. TCP/IP is a nice protocol but requires significant resources compared to “dumb” FIFO like-front-end links.
- IoT is interesting *if* one can use standard components (IoT controller targeted at < 1 USD. This is not obvious in radiation environments.
- A lot of IoT R&D is going into wireless interconnects. This *can* be interesting for future detector readouts, but it will depend on the density of “things” allowed and the amount of data achieved.
- IoT and LHC, SKA etc. . . produce “big data”, but the “data density” is very different.
- IoT is rightly very much concerned with security - for DAQ this is not very relevant.

Where channel-count is not so high and radiation is not an issue these disadvantages could likely be offset by the possibility to use off-the-shelf components and open industry standards.

There are also some common needs between IoT and large DAQ systems:

- they both create a lot of data;
- they need low-power electronics (in particular at the front-end)⁷;
- they require local intelligence for data-reduction;
- they are driven by standardization and the (increasing) use of off-the-shelf components;

IoT has already started to appear within DAQ infrastructures: the most prominent example is certainly the Board Management Controller (BMC), which allows to control electronics-boards, crates and servers in a much more fine-grained and intelligent way than was possible before. These very convenient technologies do not need more than a simple Ethernet network (wired and/or wire-less).

IoT will have for sure a strong impact on control-systems, it will drive a standardisation for the access to devices and (hopefully) obsolete a lot of legacy field-busses. The impact on specialised high-density DAQ remains to be seen and in any case for radiation sensitive detectors it remains very doubtful. In any case the “culture” of low-power, local intelligence with standardized access⁸ will be very useful for future readout-systems, in particular systems not requiring very high densities or speeds

6. Summary

Modern DAQ frameworks are based on ubiquitous industry standards, both in hard- and in software. The transporting of the data profits from its own version of “Moore’s law” and standard links will move up to the front-end where radiation and/or power permit this. Further down-stream in the data-flow tight integration with “offline” software becomes more and more important for very high throughput DAQs, because it is the only way to ensure efficient selection

⁷ Although for different reasons: IoT because it runs on battery or solar power, in DAQ the dissipation of the electronics is a major problem in particular in detectors with many fast channels in cylindrical setups.

⁸ Think of Arduino, Raspberry Pi, Beaglebone Black, . . .

and reduction of the data to manageable amounts. The DAQ in the narrow sense of acquiring and moving data around through a LAN will have practically no limits⁹. Certainly much more can be acquired than can be kept - consequently there will be the need to re-think what is kept as “raw-data”. Good DAQ frameworks are open to integrate new & emerging technologies, such as accelerators, without the need to introduce fundamental changes. Among the many technologies influencing the future of data acquisition IoT has the potential of standardizing the integration of embedded devices and it might be a driver for useful low-power, high-bandwidth wire-less technologies.

References

- [1] Adam W *et al.* 1992 *Nuclear Science, IEEE Transactions on* **39** 166–175
- [2] MIDAS Main Page https://midas.triumf.ca/MidasWiki/index.php/Main_Page [Online], last accessed Apr. 2015
- [3] DAQ-Middleware <http://daqmw.kek.jp/> [Online], last accessed Apr. 2015 URL <http://daqmw.kek.jp/>
- [4] Biery K, Green C, Kowalkowski J, Paterno M and Rechenmacher R 2013 *Nuclear Science, IEEE Transactions on* **60** 3764–3771
- [5] CHAOS <http://chaos.infn.it/> [Online], last accessed Apr. 2015 URL <http://chaos.infn.it/>
- [6] Lagin L *et al.* 2013 Status of the National Ignition Facility (NIF) integrated Computer Control and Information Systems *Proceedings of ICALEPCS2013, San Francisco, CA, USA* [Online], last accessed Apr. 2015
- [7] ITER Control System Division public homepage <http://www.iter.org/org/team/chd/cid/codac> [Online], last accessed Apr. 2015
- [8] Liu G private communication
- [9] Frank M *et al.* 2015 The LHCb Data Acquisition and High Level Trigger Architecture *21st Conference in High Energy and Nuclear Physics 2015*
- [10] Barrand G *et al.* 2001 *Comput. Phys. Commun.* **140** 45–55
- [11] Collaboration L 2014 LHCb Trigger and Online Upgrade Technical Design Report Tech. Rep. CERN-LHCC-2014-016. LHCb-TDR-016 CERN Geneva URL <http://cds.cern.ch/record/1701361>
- [12] Buncic P, Krzewicki M and Vande Vyvre P 2015 Technical Design Report for the Upgrade of the Online-Offline Computing System Tech. Rep. CERN-LHCC-2015-006. ALICE-TDR-019 CERN Geneva URL <https://cds.cern.ch/record/2011297>
- [13] Kisel I *et al.* 2015 4-Dimensional Event Building in the First-Level Event Selection of the CBM Experiment *21st Conference in High Energy and Nuclear Physics 2015*
- [14] Smith W 2013 Trigger and Data Acquisition at the HL-LHC presentation at the ECFA High Luminosity LHC Experiments Workshop

⁹ There are of course always cost-limits.