ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA042

ADAPOS: AN ARCHITECTURE FOR PUBLISHING ALICE DCS CONDITIONS DATA

J.Lång, University of Helsinki, Helsinki, Finland, also CERN, Geneva, Switzerland
A. Augustinus, P. M. Bond, P. Chochula, CERN, Geneva, Switzerland
L. M. Lechman, CERN, Geneva, Switzerland, also Slovak Academy of Sciences, Bratislava, Slovakia
O. Pinazza, INFN Sezione di Bologna, Bologna, Italy, also CERN, Geneva, Switzerland
A. N. Kurepin, INR RAS – Institute for Nuclear Research of the Russian Academy of Sciences, Moscow, Russia, also CERN, Geneva, Switzerland.

Abstract

ALICE Data Point Service (ADAPOS) is a software architecture being developed for the RUN3 period of LHC, as a part of the effort to transmit conditions data from ALICE Detector Control System (DCS) to Event Processing Network (EPN), for distributed processing.

The key processes of ADAPOS, Engine and Terminal, run on separate machines, facing different networks. Devices connected to DCS publish their state as DIM services. Engine gets updates to the services, and converts them into a binary stream. Terminal receives it over 0MQ, and maintains an image of the DCS state. It sends copies of the image, at regular intervals, over another 0MQ connection, to a readout process of ALICE Data Acquisition.

INTRODUCTION

The purpose of this article is to present an overview of a new software architecture developed for an upcoming major upgrade to the ALICE experiment at CERN, along with simulation results demonstrating its performance and stability.

The O^2 project [1] involves extensive upgrades to the ALICE experiment [2,3] as the interaction rates will increase by a factor of 100 [4], during RUN3 period of LHC (scheduled to commence in 2021). *ALICE Data Point Service* (ADAPOS) is part of the pipeline for transmitting conditions data from ALICE *Detector Control System* (DCS) to O^2, where it will be used in the reconstruction of physics data from the experiment.

ADAPOS uses Distributed Information Management (DIM), 0MQ, and ALICE Data Point Processing Framework (ADAPRO). DIM and 0MQ are multi-purpose application-level network protocols. DIM and ADAPRO are being developed and maintained at CERN. ADAPRO is a multithreaded application framework, supporting remote control, and also real time features, such as thread affinities, records aligned with cache line boundaries, and memory locking. ADAPOS and ADAPRO are written in C++14 using OSS tools, Pthreads, and Linux API.

The basic unit of *conditions data* is called a data point. A data point can contain a temperature or voltage reading, typically from a single device or channel. When the state of a data point changes, an *update (event)* is generated, which ADAPOS needs to pass on to O^2.

ADAPOS is a soft real time architecture. It consists of three applications: *Engine*, *Terminal*, and *Load Generator* (LG). According to the specifications, ADAPOS Engine needs to:

1. not lose or corrupt data;
2. preserve the order of updates to a data point;
3. handle data with throughput rate and latency as predictable as possible;
4. be stable and responsive; and
5. allow redundant instances for maximising overall robustness and maintainability of ADAPOS while avoiding unnecessary downtime.

These requirements capture the essential aspects of the architecture that were considered. As the counterpart of Engine, Terminal needs to meet the same requirements. Load Generator, on the other hand, is only a tool created for testing the performance and reliability of the other two ADAPOS applications. A manager application was also built for controlling and monitoring Engine and Terminal.

The main use cases of ADAPOS architecture revolve around conditions data. These are service subscription (i.e. obtaining conditions data), transmitting updates to the readout application of *ALICE Data Acquisition* (DAQ) (via Terminal), and publishing statistics for service quality diagnostics. Figure 1 illustrates the use cases from the perspective of Engine.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA042

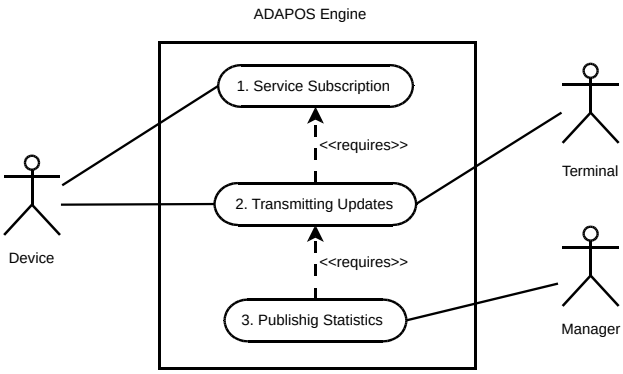

Figure 1: A UML Use Case Diagram of Engine.

ARCHITECTURE

ADAPOS applications are designed for the *CERN CentOS* [5] (Linux) operating system. C++14 was chosen to be the programming language for implementing ADAPOS, especially due to the requirement (3) for maximising the predictability of update event troughput rate. For networking, ADAPOS uses *Distributed Information Management* DIM [5] – also developed and maintained at CERN – and *0MQ* [6] protocols.

During early stages of development, the common functionality of Engine, Terminal and Load Generator was moved to a library (that gradually evolved into a more framework-like appearance, i.e. to exhibit inversion of control), known as *ALICE Data Point Processing Framework* (ADAPRO). It is open source software and publicly available at CERN GitLab [7]. ADAPRO is a software framework that can be used for building applications that can use simple configuration files and user-defined worker threads. ADAPRO threads are based on a *Finite State Machine* (FSM) model.

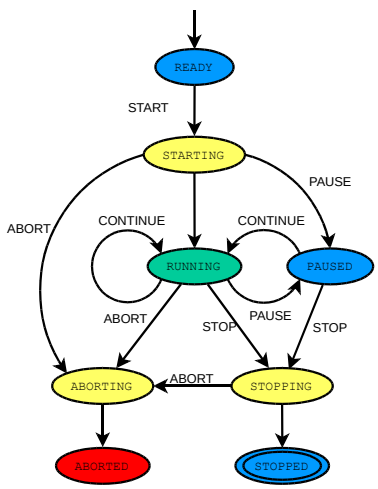

Figure 2: ADAPRO Thread States.

The way in which ADAPRO was used, by separating different computational tasks to independent and largely asynchronous threads also made ADAPOS applications meet the responsiveness requirement (4) in respect to responsiveness to commands received over DIM or POSIX signals.

Figure 3 presents an informal overview of some of the main software technologies used for building ADAPOS applications. Engine is abbreviated as 'E', Terminal as 'T', and Load Generator as 'LG'.

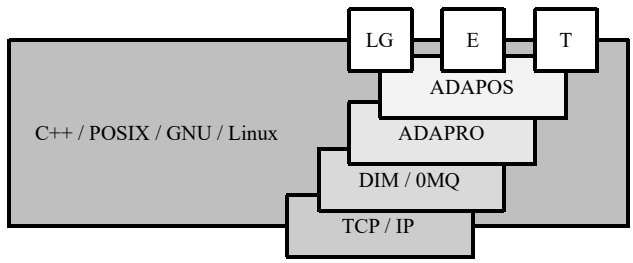

Figure 3: Software Components

Engine runs on a server in DCS network, and obtains conditions data by subscribing to the DIM services provided by the control system and its components. It stores the latest states of the data points as 128-byte binary records, known as *DataPointCompositeObjects* (DPCOM). Engine sends DPCOMs to Terminal, using a 0MQ socket.

Terminal runs on a *Front-Line Processor* (FLP), facing both the DCS and DAQ networks. It updates a contiguous array of DPCOMs, called *Full Buffer Image* (FBI). A FBI represents a snapshot of the state of DCS. Terminal keeps sending FBIs to the DAQ readout application at regular intervals, using another 0MQ socket.

Figure 4 presents a sketch showing the basic flow of conditions data in ADAPOS.

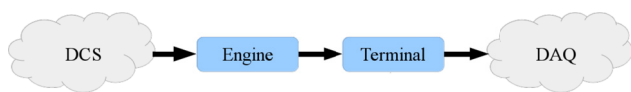

Figure 4: ADAPOS Data Flow.

LG is an ADAPRO application for Linux, that publishes data points using DIM protocol, and updates them periodically with an adjustable delay between updates. It is used for simulating conditions data sources, during the development of ADAPOS software, especially to ensure that ADAPOS meets requirements (1), (2), (3), and (4). Thanks to the design of DIM, LG is completely agnostic about the number of Engines subscribing to its services.

The reliable design of TCP, DIM and 0MQ largely ensure that the transport layer of ADAPOS meets requirements (1) and (2).

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA042

SIMULATIONS

It was necessary to run simulations in order to verify that ADAPOS meets its design requirements. The simulations also helped to determine the level of performance. During a simulation run, LG would replace DCS as the data source. At the beginning, Terminal would dump the data into a *Comma Separated Values* (CSV) file on the local filesystem. The files would then be analyzed using scripts. This solution was not scalable and the size of the CSV files became an issue.

To address the scalability issue, the generated patterns of LG were modified to grow sequentially. With this setup, Terminal could verify the updates in real time, using simple counters. Using sequential updates was also useful for verifying that ADAPOS didn't alter the order of updates, thus meeting requirement (2). Whether the data in update events grows sequentially or is random doesn't affect integrity, which is a fact also verified in a different set of simulation runs.

The main performance aspects tested during simulations were the number of LG instances publishing updates simultaneously, the number of services published per LG, throughput rate of updates, i.e. the number of DPCOMs processed per second. It was also noticed that the Linux scheduler had a major performance impact, especially on the stability of the sustained througput rate. Thus, in the later simulations, *numactl* and/or *POSIX Threads* library were used successfully to stabilize the throughput rate, thus making ADAPOS fulfill requirement (3).

After hundreds of hours of different simulations on a hardware that matches the FLPs that are going to be used during RUN3, results on ADAPOS performance were obtained. The obtained results provide good indication on the level of performance of ADAPOS.

RESULTS

The simulations helped to optimize the performance and stability of ADAPOS software. Figures 5, 6, and 7 present the results of the 33 10-minute simulation runs with ADAPOS 1.0.0, during which no event loss or corruption was detected. The coloured bars present the average throughput of each simulation run. The grey error bars represent the minimum and maximum measured throughput during the simulation run, so they give some idea about best case and worst case performance.

During the nine simulation runs used for producing Fig. 5, thread scheduling parameters were not adjusted, and the LGs were configured to add a 5 µs delay between updates.

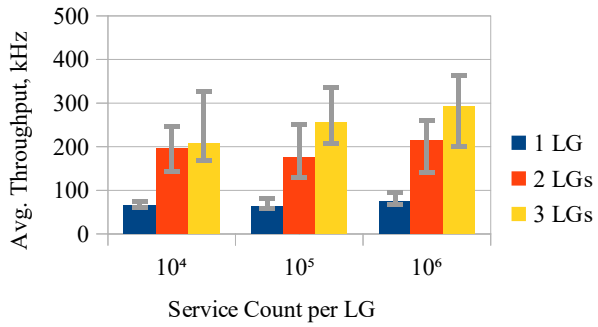

Figure 5: Simulation Results, Part I.

Figure 6 shows results for simulation runs where 5 µs delay between updates were used. Every ADAPOS process was started with numactl -N 1 -m 1.

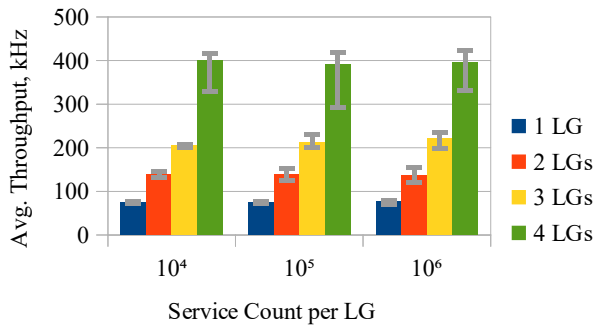

Figure 6: Simulation Results, Part II.

Figure 7 shows results for simulation runs using 1 µs delay and numactl -N 1 -m 1.

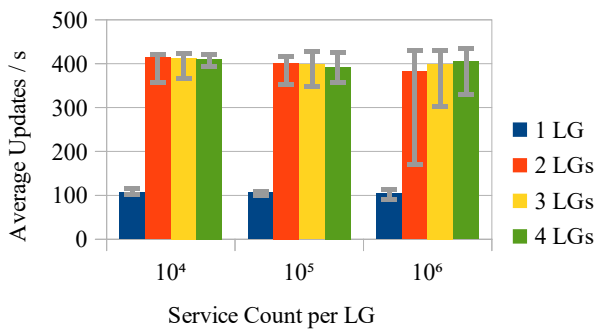

Figure 7: Simulation Results, Part III.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUPHA042

When operating below the maximum capacity, the performance of ADAPOS was found out to scale roughly linearly, with a notable exception that seems to be related to thread scheduling.

The number of data points per LG don't seem to have effect on performance, except for increasing the startup time of Engine proportionally. When starting Engine with the number of services being in the order of thousands, the startup time was in the order of milliseconds, whereas starting Engine with millions of service, subscriptions took several minutes. The expected number of services in production is around 100 thousand services, so the startup time is not an issue.

The maximum level of performance with the test setup was found to be around 400 kHz, which is two orders of magnitude better than the current requirements. When a certain stress level is exceeded, ADAPOS reaches its maximum performance, which shows in Fig. 7. By contrast, in the simulations associated with Fig. 6, the throughput was still growing linearly as a function of the number of LG instances. In the test setup, the Linux scheduler started assigning dedicated CPU cores to ADAPOS threads when their CPU time to real time ratio gets high enough.

Because ADAPOS applications only keep track on the latest known state of each data point, the same memory allocations can be reused. Thus, the memory consumption of ADAPOS applications doesn't increase over time, unless there are recursive memory leaks. *The simulations confirmed that the memory usage remained constant, even during a span of several days of maximum operation at maximum throughput at a time*, with one important exception. It appears that *the size of the 0MQ input queue of Terminal can start growing very fast if Terminal processes updates slower than Engine*. Thus, monitoring the memory usage of Terminal can be very useful for diagnostics.

As stated above, *there were no event loss or problems with data integrity detected while running the simulations included in this article*. Stress tests that lasted for days on maximum performance also indicated no signs of data loss or corruption.

CONCLUSION

ADAPOS turns out to meet its specifications.

It was found out, that the Linux scheduler has a major impact on ADAPOS performance, especially on a multi-socket system, when it moves threads from one CPU socket to another. Because ADAPOS is an input-output oriented system, performance improvements are hard to achive by compiler optimizations like loop vectorization. Instead, adjusting scheduling, paging and other such runtime system parameters, is likely to yield speed boosts more easily.

ADAPOS 4.0.0 and newer versions also support the usage of two redundant Engines to cope with requirement (5) and the mechanism is known to work, based on later simulations. However, the use of multiple Terminals or Engines that subscribe to different services, remains yet to be implemented and tested. Also, adding support for an ADAPOS configuration database, may become a future priority. Furthermore, developing methods for detecting when ADAPOS exceeds the maximum level of stable performance, remains an option for future research and development.

ADAPOS also serves as a case study for demonstrating the features of ADAPRO. Ideally, adjustments in thread scheduling could be implemented in ADAPRO framework itself, making it more attractive tool for general audience in real time parallel programming or physics control systems. Other possible future improvements include an unified asynchronous stream-like API to DIM and 0MQ access and a simpler thread model in ADAPRO.

ACKNOWLEDGEMENTS

I'd like to thank all the coauthors for their help on this project. Also, I'm grateful to the *Finnish UNIX Users Group* (FUUG), who gave me a grant for participating in ICALEPCS 2017.

REFERENCES

[1] The ALICE Collaboration. Upgrade of the online-offline computing system, CERN-LHCC 2015-006, June 2015

[2] The ALICE Collaboration, "The ALICE experiment at the CERN LHC", JINST 3, S08002, 2008.

[3] The ALICE Collaboration. Upgrade of the ALICE Experiment: Letter Of Intent. J. Phys. G 41 (2014) 087001

[4] The ALICE Collaboration. Upgrade of the ALICE Readout & Trigger System. CERN-LHCC-2013-019, September 2013.

[5] C. Gaspar, M. Donszelmann, "DIM –A distributed information management system for the DELPHI experiment at CERN", Proceedings of the 8th Conference on Real-Time Computer applications in Nuclear, Particle and Plasma Physics, Vancouver, Canada, June 1993.

[6] 0MQ, http://zeromq.org/.

[7] CERN GitLab repository for ADAPRO, https://gitlab.cern.ch/adapos/adapro