Heterogeneous data-processing optimization with CLARA's adaptive workflow orchestrator

Vardan Gyurjyan[1,*], *Sebastian* Mancilla[2]

[1]Thomas Jefferson National Accelerator Facility, Newport News VA, USA
[2]FSMT University, Valparaiso, Chile

Abstract The hardware landscape used in HEP and NP is changing from homogeneous multi-core systems towards heterogeneous systems with many different computing units, each with their own characteristics. To achieve maximum performance with data processing, the main challenge is to place the right computing on the right hardware. In this paper, we discuss CLAS12 charge particle tracking workflow orchestration that allows us to utilize both CPU and GPU to improve the performance. The tracking application algorithm was decomposed into micro-services that are deployed on CPU and GPU processing units, where the best features of both are intelligently combined to achieve maximum performance. In this heterogeneous environment, CLARA aims to match the requirements of each micro-service to the strength of a CPU or a GPU architecture. A predefined execution of a micro-service on a CPU or a GPU may not be the most optimal solution due to the streaming data-quantum size and the data-quantum transfer latency between CPU and GPU. So, the CLARA workflow orchestrator is designed to dynamically assign micro-service execution to a CPU or a GPU, based on the online benchmark results analyzed for a period of real-time data-processing.

1 Introduction

We are witnessing unprecedented expansion of scientific data. This is explained by upgrades to existing accelerator facilities, such as LHC and CEBAF, as well as improvements in detector equipment. As a result, data volumes are increasing and thus, putting pressure on computing (both hardware and software) environments. It is expected that the traditional software and hardware infrastructures used in HEP and NP will have difficulties processing large amount of data generated by new experiments [1]. To address ever increasing data requirements of contemporary and future experiments, new data processing applications must be able to scale within a multi-core CPU environment and utilize an increasing number of heterogeneous HPC platforms including accelerators such as GPGPUs, TPUs, FPGAs, and tiered memory systems. An efficient operation on these platforms will require a tight integration of components handling the data flow, distribution and processing, as well as workflow management. Data-processing application development frameworks must be able to integrate data processing components that internally use different parallelization models, such as physics generators and detector simulation. They must encourage common development and code reuse by creating methods to integrate new developments with decades of legacy work across experiments.

[*] Corresponding author: gurjyan@jlab.org

1.1 CLARA: CLAs12 Reconstruction and Analysis framework

The CLARA is a data-in-motion platform to build streaming scientific-data analytic applications [2]. It enables scientists to collect, process, analyze, and act on streaming data-quanta (user defined and framework agnostic) across diverse hardware and software infrastructures. The CLARA framework presents three core components (see Fig.1): data processing station (a micro-service, that provides runtime environment for user data-processing engines), data-stream pipe (a data-bus, that supports protocols, such as MPI, pub-sub, p2p, inproc, and POSIX shared memory), and a data-flow Orchestrator (a process level workflow management system).

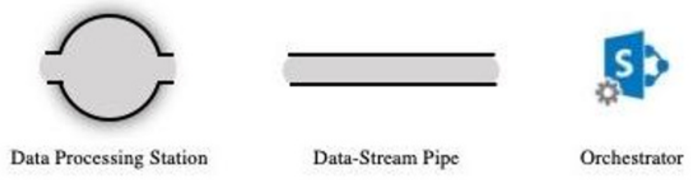

Fig. 1. CLARA core components

The idea is that the stream of data-quanta is flowing through data processing stations and getting changed/processed along the line of flow. Data processing stations are the place where user data-quanta processing algorithms and programs are dropped in (as shared objects or dynamically loadable classes) and presented as data processing micro- services.

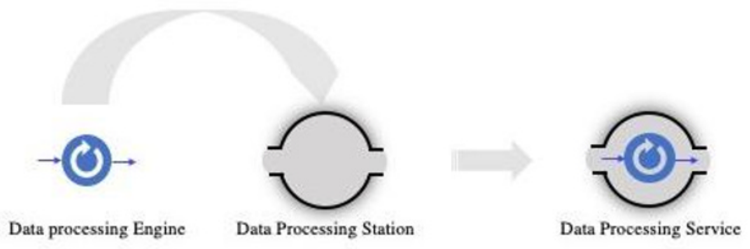

Fig. 2. CLARA micro-service

Micro-services in CLARA are reactive computing units. They react on a data-quantum at the input and present processed data-quantum at the output. Data processing application in the CLARA is composed of interlocking micro-services. Micro -services are linked to each other by the data -stream pipe and are hiding technology (e.g. CPU accelerator hardware technologies, HL software languages, etc.) and algorithmic solutions used to process data. Once a given service receives a valid data quantum, that service executes its engine, producing a new data quantum, and passing it to the next service in a data-flow path. Services can be deployed and connected together at start time or at run-time. Micro-services publish their data processing functionality details to the Registrar normative service of the framework that can be probed by the application orchestrator to discover and dynamically deploy a desired micro-service in an active workflow.

2 Data-flow through micro-services

Clara micro-services are reactive in nature, designed in a flow-based programming paradigm [3]. This framework is using reactive, micro - services design to build asynchronous processing pipelines. It is an event-based model, where data is pushed to the consumer micro-service as it becomes available. In this paradigm, micro-services react on an asynchronous sequence of events. A reactive micro-service adds the concept of operators, which are chained together to express what processing to apply at each step to the data. Applying an operator returns a new intermediate publisher. In fact, CLARA's micro-services can be thought of as both a subscriber to the operator upstream and a publisher for downstream. The final form of the data ends up in the terminal micro-service that defines what to do with the data from a user perspective. The terminal services are usually data persistency and data visualization micro-services.

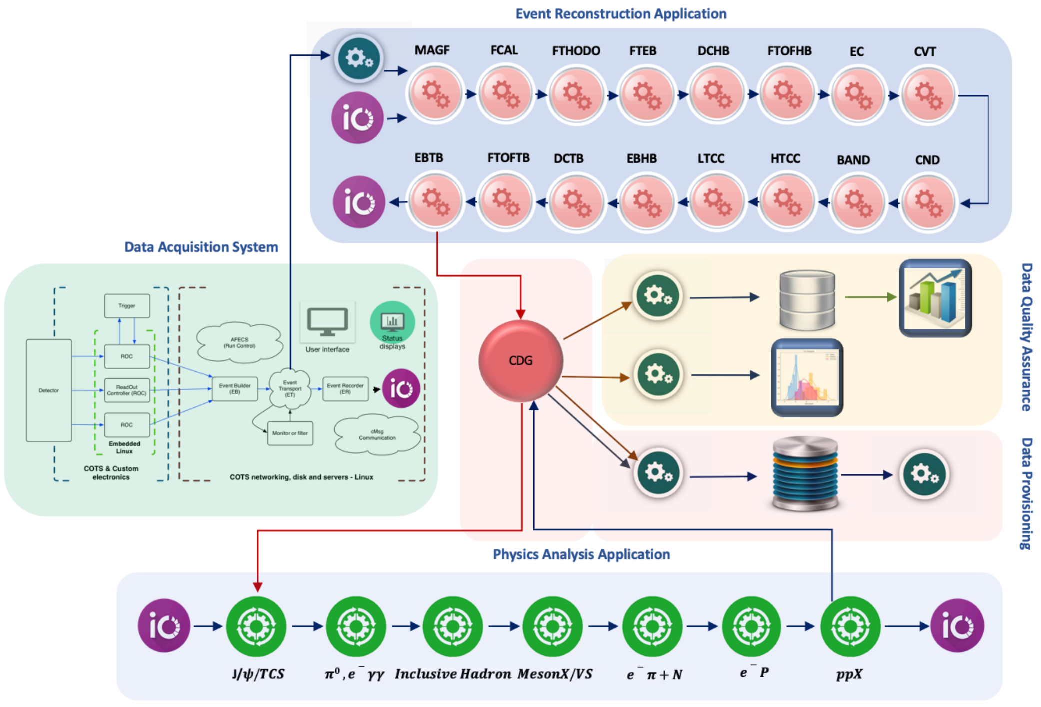

Fig. 3. CLAS12 data processing workflows

Figure 3 shows data flow diagram for the CLAS12 data processing applications [4], including data acquisition, event reconstruction, data quality assurance, data provisioning and data analysis. Data acquisition system is not based on CLARA and is presented here for the purpose of showing that entire CLAS12 offline data processing pipeline is also used to process streaming events off the DAQ system. Note that the event reconstruction application micro-services composition (see Fig3) is for event level parallelization, where all the components of a single event are reconstructed sequentially, while multiple events are processed in parallel. However, micro-service environment allows us to perform sub-event level parallelization without additional programming efforts. Simply by changing the reconstruction application service composition, (presented to the application designer as a YAML text file) the group of micro-services will reconstruct the same event (different parts of it) in parallel. Reconstructed data is being accessed through CLARA data ring (CDG), which is in-memory data-hub with configurable/optional event retention depth. On the button of the Figure 3 is a data pipeline for CLAS12 data analysis trains, having micro-services to analyze reconstructed data for a specific physics of interest. Analysis trains can run during online data taking, during data cooking processes, or off the persisted cooked files.

3 Process level workflow management

CLARA presents a default workflow manager for each and every data processing application. This workflow orchestrator plays the role of an event-based (or streaming unit-based) workflow management system to guarantee a proper execution and monitoring of a data processing application. Also, CLARA provides workflow orchestration API to build and/or modify custom workflow management systems. Default workflow orchestrator is responsible for the application service-composition, including micro-service discovery, deployment, configuration, linkage, and application execution. It is important to mention that the workflow orchestrator is not interfering with streaming data-processing and assumes a passive observer role during the data processing, monitoring (by subscribing reporting messages from micro- services), and performing a real-time benchmarking for each micro-service. It logs any exception and info messages it receives. CLARA can operate in a batch mode, where orchestrator auto -generates and executes batch or cloud deployment scripts (supports, PBS and SLURM). An orchestrator will optimize service deployments in real time to achieve optimum performance. It also detects available data processing environments (computing resources), and if required will elastically expand and/or reduce the running application. For example, if new nodes become available on the farm, an orchestrator can horizontally scale application, or in case one of the active processing nodes goes offline, it will continue the application with reduced number of micro-services. An orchestrator also handles data-at-rest provisioning by staging it on a node local file system to optimize IO performance. The CLARA orchestrator is the front-end of data processing that is presented to a user in the form of a rich command line interface (CLI).

3.1 Heterogeneous workflow optimization

Micro-services are units of extremely low coupling. They communicate with each other with low dependency (data only). This model presents an ability to build hierarchical compositions or parallel data processing pipelines, that can span over heterogeneous hardware/software infrastructures. To improve the performance of CLAS12 data analytics, (primarily the tracking micro-service engine: the slowest component in the CLAS12 data processing pipeline), a new hit-based tracking engine, running on GPGPU was developed. Currently this service is in a testing and validation stage, showing dramatic performance gains. In the near future, JLAB farm will include GPUs and maybe other accelerator augmented nodes. So, we will need to track service affinity or determine which type of resource offers the most suitable cost-performance trade-off for a particular micro-service. Accounting for the heterogeneity in the farm becomes more difficult when a cluster is shared among multiple jobs (the main operational mode at the JLAB farm). So, the CLARA workflow management system (orchestrator) is equipped with adaptive functionalities to guarantee optimum performance, and minimize underutilization of a resource in a shared heterogeneous cluster. Hence, knowing service/hardware-instance type relationships (which we call service affinity), and finding a good mix of different hardware allocations becomes critical. Job submissions resulting in hardware allocation has a limited predictability. Users submit a job to the farm with a few hints about the job characteristics, including memory requirement and ability to use special accelerators, like GPU. The CLARA workflow management system is programed to define service affinity using real-time calculated relative computing rate (Fig 4).

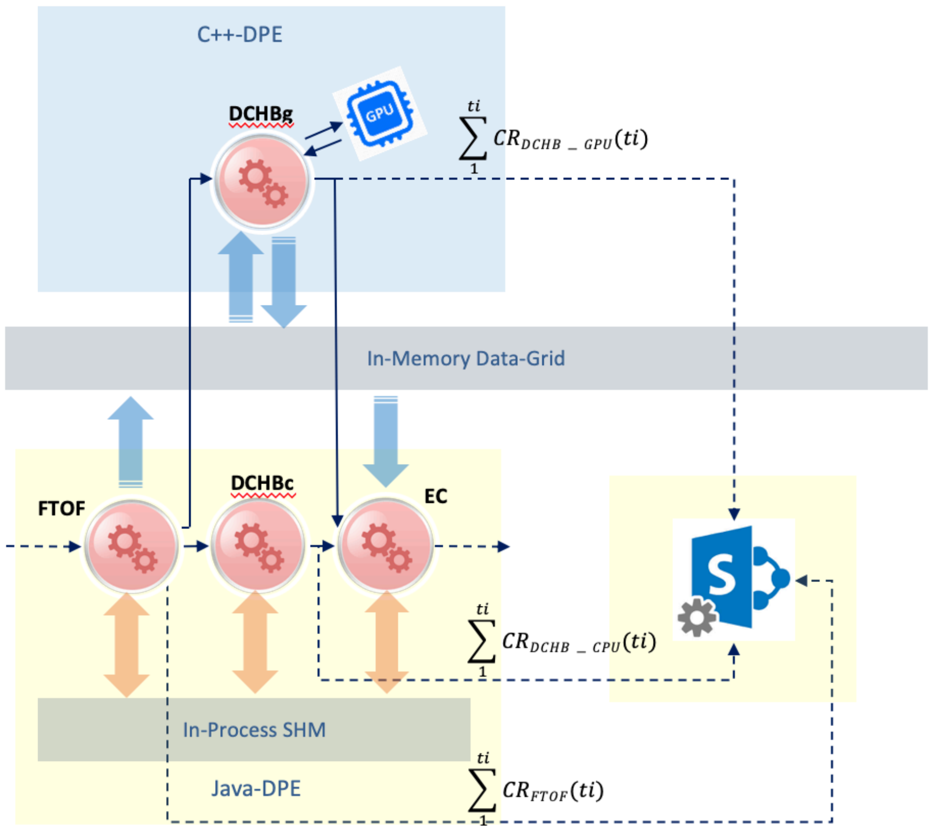
C++-DPE
DCHBg
GPU
$\sum_{1}^{ti} CR_{DCHB\_GPU}(ti)$
In-Memory Data-Grid
FTOF
DCHBc
EC
$\sum_{1}^{ti} CR_{DCHB\_CPU}(ti)$
In-Process SHM
Java-DPE
$\sum_{1}^{ti} CR_{FTOF}(ti)$

Fig. 4. CLAS12 heterogeneous dataflow deployment

For e.g. counting rate for the forward time of flight micro-service is a time spent for processing a single event on a single core/slot. The sum of available processing slots t_i indicates performance of a particular micro-service, scaled vertically over the t_i slots. In this shown scenario, we have access to 2 nodes (see Fig. 4), one being equip with an accelerator GPU. At the deployment stage, the CLARA orchestrator or workflow manager will deploy 2 hit- based tracking micro-services in the composition, streaming events to both DCHB (Drift Chamber Hit Based) micro-services in the pipeline. So, after running *n* events FTOF (Forward Time of Flight), DCHB (CPU based), DCHB (GPU based) will report computing rates to the orchestrator. At that point, the orchestrator will calculate process share variables:

$$P_g = \frac{\sum_1^{ti} CR_{FTOF}(ti)}{\sum_1^{ti} CR_{DCHB\_GPU}(ti)} \quad \text{and} \quad P_c = \frac{\sum_1^{ti} CR_{FTOF}(ti)}{\sum_1^{ti} CR_{DCHB\_CPU}(ti)}$$

for both CPU and GPU branches of the pipeline, and will decide which branch to keep for the current production. Due to the fact that in the farm deployment there are many moving parts, such as data serialization, data copying, job landed hardware performance, etc., and flexibility provided to users to change/update/redeploy engines, this makes ahead of time deployment optimizations unrealistic.

Summary

We developed an event-driven, reactive data-stream processing framework that implements a micro- service architecture and a flow-based programming paradigm. This model presents an ability to build hierarchical compositions or parallel workflows across diverse hardware and software structures. In this environment, the ability to compose data processing application through service compositions/workflows rapidly and with less effort than through general programming will be of substantial value. The decoupled nature of CLARA micro-services allows us to design hybrid data processing applications, along with contemporary accelerators with appropriate data-flow optimizations to efficiently handle the massive parallelism and heterogeneity of modern computing facilities. The CLARA framework is used in CLAS12 as the main data-processing framework to design and operate event reconstruction, data analysis, and calibration, as well as data distribution and data visualization applications. The framework is also used outside of JLAB, namely at NASA [5] for MODIS and SCIAMACHY satellite data fusion and processing applications.

References

1. "A Roadmap for HEP Software an Computing R&D for the 2020s", arXiv:1712.06982v3 [physics.comp-ph] 11 Feb 2018
2. "CLARA: A Contemporary Approach to Physics Data Processing", 2011, J. Phys.: Conf. Ser. 331 032013 doi:10.1088/1742-6596/331/3/032013
3. "Reactive Microsystems" by Jonas Boner ISBN:978-1-4919-9435-1, 2017 O'Reilly Media, Inc.
4. "CLARA: The CLAS12 Reconstruction and Analysis framework", 2016, J. Phys.: Conf. Ser. 762 012009 doi:10.1088/1742-6596/762/1/012009
5. "Earth Science Data Fusion with Event Building Approach", 2015, IEEE DOI: 10.1109/BigData.2015.7363972, ISBN: 978 1-4799-9926-2