Strategy concept for identifying highly complex (multi-parametric) signals in huge data streams

TA5

January 2023

Contents

To-Do

1. Distribution of work - contributions to sections

2. Adding of definition and scope as introduction

1 Challenges in data irreversibility

Michael

The rapid increase in both data rates and data complexity leads to several challenges soon to be seen throughout society as we enter the "Internet-Of-Things" era, where large sets of "sensors" will transmit data upon which autonomous "actors" will react. However, the substantial increase of power consumption for storage solutions, e.g. cloud computing, requires the investigation of resource-optimised data sets with maximal relevance and minimal redundancy. Decisions will need to be made, often in real-time and without human intervention, which information to keep or how to compress it with calculable loss. Loss will be inevitable and mostly irreversible, while off-line analyses or emerging additional information will feed back and dictate modifications of the on-line processes ("dynamic filtering").

A common challenge is the identification of a signal in huge data streams. The challenge does not only arise from the amount and velocity of the data, demanding on-the-fly decisions on huge data volumes, but also because the filtering needs to be robust. Real signals must not be missed, while large false-alarm rates will overwhelm the system downstream. In some experiments, e.g. in astrophysics, one may want to trigger measurements with other instruments, which can be costly if false signals are followed up too often.

Often it is useful to distinguish between expected and unexpected signals. Sometimes, the experiment may need to be designed to look for and isolate a signal that is expected. In contrast, discovery sciences make some of their largest progress by experimentalists stumbling over serendipitous signals that had not been seen before. Ideally, an experiment should be designed to deal with both, even though a varying noise background, or the presence of significant "intereferers" may need to be accounted for. Large data volumes and rates make this increasingly more challenging.

In any case, the decision process of rules and methods for the extraction of pertinent information out of huge data streams in real-time will need to be updated frequently and captured as important metadata. Hence, the impact of the information loss must be traced and gauged in order to allow drawing adequate conclusions from archives, which will no longer be static but dynamic entities.

Against this background, this document aims to develop a concept of how to tackle the identification of highly complex (multi-parametric) signals in huge data streams. As work in TA5 progresses, the contents of the document will be reavaluated and modified.

1.1 Data irreversibility and dynamic metadata

In the case of on-line filtering or processing, only a (limited) version of the original data can be stored. Sufficient metadata then need to be provided on

how this selection process was implemented and executed. This is crucial to evaluate the content of an archive, to present new choices of criteria for online filtering or to judge the discovery potential in light of selections made. Therefore, in addition to metadata that describe the basic properties of the data and the instrumentation used, we need to provide additional metadata that describe the first selection of data. Especially, we need to capture in some way a description also of why or which data have *not* been captured. We need extra metadata including the complete chain of algorithms needed to enable, in principle, the reproducibility of selections.

In a dynamical life cycle of data (see Figure 1), the filtering process is not static but dynamic. Any scheme must be flexible enough to accommodate different types and numbers of decision processes.

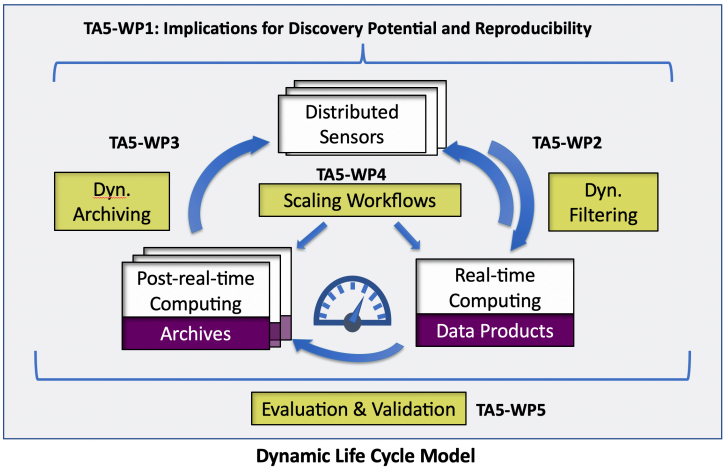

Figure 1: Structure of TA 5 as dynamic life cycle of data

A detailed description of the tight interconnection between all work packages (WPs) of TA5 has been presented in the corresponding section in the PUNCH proposal [1]. Our concepts and developments aim at common solutions for processing of dynamic metadata in the fields of PUNCH sciences and potentially beyond. Here it is important to note one main distinction between high energy physics experiments and astrophysical observations: In collider experiments of high energy physics one can practically increase the number of metadata by improving and repeating experimental setups. However, this is not the case for astronomical observations where the observed object, the universe, is literally out of control. One can only partially overcome this problem by turning to statistically large samples of objects. In contrast, in large data streams originating from smart cities, for instance, a similar situation occurs, ie. the "experiment" cannot be controlled, but large samples are also not available. Developing techniques for such situations are among the intended tasks for TA5, but not in

scope of this document.

In a general sense metadata describe other data thus supporting the F.A.I.R. principles in a direct way. In the broad context of NFDI, metadata are information (context) that describes an object such as a dataset as richly and complete as possible. As data is further analysed, processed or related to other data, metadata tend to grow in time. In other words, additional layers of metadata will be added, corresponding to the increasing levels of metadata

Add reference to metadata document.

1.2 Requirements for future astrophysics applications

Astrophysics deals with two different types of large data volumes, observations and simulations. Improved technology, larger number of detectors, digitizing wider bandwidth, or sampling larger portions of the sky at different parts of the electromagnetic spectrum all lead to many more data. The data are useless until they are analysed, understood and often matched or compared with other types of data. Often they are compared to simulations, which are becoming huge in volume in their own right.

Even when ignoring the even the data management aspects, the data (observational and simulations) need to have description of how they were created or derived. Data format that allow a comparison or inter-operational aspects have been established in astrophysics for a long time. What is needed is a scheme that allows to identify and describe the filtering mechanism. This needs to become part of the dynamical archive, so that the user can formulate meaningful queries and influence future experiments by being able to modify the filters.

Astrophysics requirements therefore exist on the experimental side and the archiving side. For the experiment, technology to cope with the data, to apply the filters and to save the identified fraction of the data together with adequate meta-data. The meta-data by themselves are likely to have a considerable size, and data and meta-data need to be digested into an archive that is flexible enough to accept entries that may adapt their structure from experiment to experiment.

1.3 Requirements for high energy physics experiments

Andreas

Efficient data selection in high energy particle and nuclear physics also necessitates more complex decisions in significantly less time intervals. Technically, that challenge can be partly solved by further developing software-based triggering processing detector data in parallel where possible. Recently the LHC experiments ALICE and LHCb have demonstrated that triggering in software enables the selection and storage of specific complex events of interest.

However, more generic searches targeting at unexpected, anomalous signals require more flexibility in the triggering process. Trying to store anomalous signal would be also a change of the established paradigms of a hierarchically structured scheme of triggers. A classification of non-signal like events as opposed to statistical backgrounds will probably require a sophisticated use of neural networks in online workflows.

Another option supporting complex and more advanced selection of events can be related to information from external data bases. Querying such archives efficiently could allow to trigger on dedicated signatures that might be related to other results of potentially new physics from related analyses of PUNCH data volumes.

2 Technologies for data-intensive processing

The sorting and selection of detector data at the earliest stages is a crucial aspect of real-time data analysis. To this end, key algorithms for real-time data reduction, including massively parallel sorting and selection algorithms have to be optimised for their usage in massively parallel systems. As an example clustering, i. e. the reconstruction of localised features requires generic, massively parallel clustering algorithms that are relevant in several fields of PUNCH sciences. As an intermediate step, both algorithms and underlying data structures have to be further developed to meet the requirements of the necessary scalability. Corresponding design principles should therefore lead to data and/or task parallelism with a focus on performant algorithms and data structures.

Furthermore, new trends based on the processing of data streams in FPGAs, programmed with input from machine learning are promising technologies for developments.

2.1 Requirements for massively parallel processing

Vadim

Ongoing and future experiments in astrophysics and high energy physics will produce data that should be treated as Big Data in terms of the 3 Vs: volume, velocity and variety. The volume refers to the amount of the produced data. But not only the volume is challenging, the speed with which data are being generated, and as a result, the speed with which the data should be processed as well as its complexity require new approaches in data reduction.

The IT-infrastructure used for the big data handling is thought to be horizontally scalable at least because of the large volume of the data to be simultaneously processed. A paradigm that allows the computing environment to be scaled out in a nearly unlimited way is Massively Parallel Processing (MPP). The main feature of this architecture is that the computing nodes do not share memory or any other resources interconnecting over a high-speed network (Fig. 2). This imposes a fundamental limitation on the range of problems and used algorithms, especially those related to machine learning. While many parallel algorithms are efficient within an infrastructure with shared memory, they suffer from scalability problem when being applied in MPP. Since the communication between nodes in MPP is a principal bottleneck, in order to leverage this computing model, the problem should be divided into independent and non-iterative tasks where data exchange between nodes is as little as possible.

The MPP architecture is already widely used in the enterprise sector, with different requirements and goals though. In future scientific experiments, how-

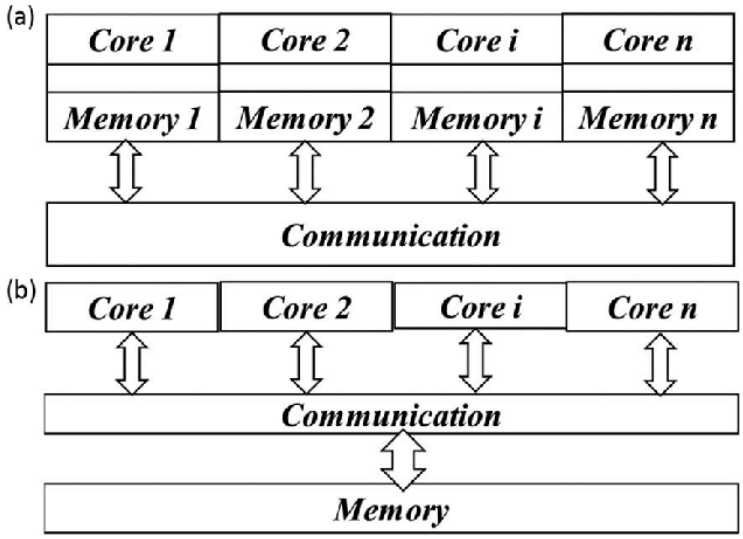

Figure 2: Parallel computing models: (a) distributed memory (MPP), and (b) shared memory. Credit: [2].

ever, the amount of generated data is expected to be larger (e.g. up to 3 Exabytes/year in the SKA experiment) and the data reduction is much more sophisticated.

The sorting and selection of detector data at the earliest stages is a crucial aspect of real-time data analysis. To this end, key algorithms for real-time data reduction, including parallel sorting and selection have to be optimised for their usage in MPP systems. Both algorithms and underlying data structures have to be further developed to meet the requirements of the necessary scalability. From a practical point of view a design that helps avoiding data races also for workflows with asynchronous execution is needed. Technically, synchronisation at different levels of threads and blocks is possible and discussed below.

In this context, clustering is one of the approaches that can be utilized for object detection and selection in the incoming data stream, i.e, allowing in fact data filtering, which is relevant in several PUNCH-related science fields. There are a variety of traditional clustering algorithms, such as K-Means, EM-Algorithm, DBSCAN, BIRCH, Spectral Clustering, etc. Depending on the data set structure and specific task they can achieve good results as long as the data set is small enough, i.e., when it fits into the memory. The traditional clustering algorithms thus suffer from the scalability problem and do not meet the needs of big data clustering. Parallel clustering algorithms divide data into chunks being distributed over the nodes where they are simultaneously processed. A parallel algorithm succeeds if the individual tasks are not-interactive and independent as much as possible.

Recently, there have been attempts to expend the use of different clustering algorithms in MPP.

The most popular clustering algorithm, because of its simplicity, is k-means. While this is an iterative algorithm, which makes it challenging to apply in MPP, several attempts have been made to obtain high performance by eliminating the iteration dependence. In [3] it is proposed to estimate the iterations by sampling of the original data set to obtain only some subsets to use their centers to cluster the original data set. Another approach is suggested in [4], where the iterative k-means processes are run concurrently with multiply centroid groups, then low-quality processes are pruned and new ones are started, keeping the best solution at the end. This and Other parallel approaches for the k-mean algorithm are mostly based on the MapRedice distributed computing model, which is also an MPP-like model. An interesting clustering algorithm for peer-to-peer unstructured networks, i.e. nothing to share, based on K-medoids and k-means algorithms has been introduced in [5].

Another popular clustering algorithm is DBSCAN. A MapReduce-based variation of this algorithm is proposed in [6]. More advanced implementations of the DBSCAN algorithm have been proposed in [7, 8] and [9]. An implementation with nothing sharing based on MPI is proposed in [10], where the authors also adopted the Single Instruction Multiple Data (SIMD) technique.

There are many other clustering algorithms that can be parallelized within the MPP paradigm (e.g. [11]). An overall vision of the current state of this topic is provided in a recent review [12].

In should be noted that the MPP paradigm has various implementations. The most straightforward and therefore popular, especially in the enterprise area, is the MapReduce technique [13]. A similar technique is developed in the Spark framework [14], though the MapReduce technique itself does not depend on a specific software or hardware platform. Spark is widely used because of its high-level interface allowing to perform big data analysis with minimal efforts, i.e. it is petty user-friendly. The cost of that usability is a lower performance compared to other tools with a lower-level interface such as the Message Passing Interface (MPI). In some cases, when dealing with non-optimized ML algorithms, especially including matrix algebra, programs written with the MPI outperform Spark by tens of times in speed and memory consumption (see, e.g. [15, 16, 17] and [18]).

2.2 Multi-core CPU

Vadim

In recent years the scaling of transistors on microchips tends to fall below the increase predicted by Moore's law. Conversely, compute capability has been significantly enhanced by adding more CPU cores (the multicores architecture) or using GPUs (see below). Modern multi-core CPUs are optimised for low-latency access to cached data sets. They provide a control logic for out-of-order and speculative execution. It is interesting to note that performance gains through vectorisation using Single Instruction Multiple Data (SIMD) instructions, and parallelisation using many-core CPU architectures often require less significant

changes to the code base as would typically be required for porting algorithms to GPUs. The latter often requires specialized code redesign and further optimisation, as also outlined in the subsequent section. The SIMD instruction model also implies that vetorised instructions on modern CPU SIMD cores are executed in lockstep. Therefore no synchronisation barrier is needed, because all elements of the corresponding vector finish processing at the same time.

2.3 GPUs and hybrid architectures

Andreas

Modern GPUs are optimised for data-parallel, high-throughput computation. An efficient usage of GPU acceleration is crucial for many (upcoming) experiments in PUNCH sciences. Evaluations of real-time reconstruction at LHC detectors [19] have shown e.g. that high compute loads for parallelisable tasks are ideally suited for GPUs, so that they are a quite general option. Achievements in GPU-accelerated processing and reconstruction in big data experiments are documented in a number of publications, e. g. ALICE [20, 21] or CBM [22] and LHCb [23]. Also in the context of A Common Tracking Software (ACTS) the utilisation of GPU-based Kalman Filter for Track Fitting has been investigated in [24].

Because the performance gain of a GPU is mainly based on the ability to run thousands of threads in parallel, this limits considerably the amount of memory available per thread compared to the CPU. Technically, a typical GPU is added to a server by means of a PCI-e card, and the corresponding bandwidth of the bus is limiting the possibilities of an efficient use of the GPU. In a versatile trigger approach, where GPUs can be used for several applications, the use of the GPUs needs a careful planning in the trigger software to optimise the performance of the system. Therefore, algorithms aiming at an optimised load balance have to be developed, deriving the decision where to execute a certain algorithm on the current and predicted load of the nearby CPUs and GPUs. A promising approach to further optimise the trigger performance is to make use of the possibility to invoke hard/software co-design issues based on the new type of CPUs which include GPU or FPGA features. The load balancing algorithms will then be augmented to include new hardware to dynamically optimise the performance of this new compute platform.

It is interesting to note that models of Single Instruction Multiple Threads (SIMT) can describe workflows on GPUs. In this case threads are used instead of vector registers. GPUs consist of multiple processing elements, each with multiple SIMT GPU cores. As a result, not all threads are processed in lockstep leading to the requirement of a corresponding synchronisation instruction.

2.4 FPGAs

Arno

Field Programmable Gate Array (FPGAs) have become everyday tools in modern digital processing systems. They provide real-time and highly parallel processing of large data streams. High-speed serial links are directly integrated in the device in order to receive and transmit the data efficiently. The total data

throughput rate can easily reach more than 1 Tbps and is thus a main advantage of FPGAs compared to CPU or GPU based systems. The basic data processing is performed by logic elements and digital signal processors. System-on-chip devices also integrate an ARM processor and possibly high-bandwidth memory. Dedicated FPGAs targeting machine learning applications include AI engines for fast vector processing attached to local memory which may be connected to a network-on-chip.

In detector and measurement systems, FPGAs are usually employed right after the signal digitisation stage where the full raw data is available. Typical tasks are conventional digital signal filtering, anomaly detection, feature extraction and triggering of interesting events. For particle physics applications, clustering and tracking algorithms are e.g. implemented on FPGAs as part of the feature extraction. With the processing resources available in modern FPGAs, machine learning applications and artificial neural networks can be implemented on FPGAs. Software and design tools are however needed to convert the trained neural networks into efficient FPGA implementations. These may perform direct conversion from neural network training result into VHDL, or make use of High Level Synthesis (HLS) tools. The optimisation of the implementation needs to take into account the boundary conditions given by the measurement system, like maximal latency or the required data bandwidth, and balance against the available FPGA resources. In case data reduction is the goal of the processing algorithm, monitoring processes may collect summary information from the data which complements the extracted core features and can be stored as meta-data. FPGAs are thus powerful and flexible platforms for high-bandwidth data processing, feature extraction and data reduction.

add references

2.5 Machine learning

add existing examples and a few concrete next steps

Arno

Machine learning is ubiquitous in today's data-driven research. In particular, artificial neural networks have been introduced to all stages of data processing in astrophysics and high-energy physics experiments. In the context of this document, data reduction, feature extraction and event categorisation in very early data processing stages are of interest. While high-energy physics may have simulation models that can be used for network training, astrophysical applications often must rely on detection of anomalies from the regular data patterns. Supervised and unsupervised learning must therefore both be explored to obtain best signal-to-background ratio. Convolutional and recurrent networks are typically applied in order to perform feature extraction tasks, like identification of certain physics event types or reconstruction of amplitude or time of detector signals. Another example are auto-encoders which may be applied for anomaly detection tasks. Plenty of applications already exist.

add references

The preparation of the input data to the machine learning tools is an important step. Normalisation, scaling or decorrelation may be necessary to obtain good training results. Software tools for defining the network architecture and for performing the network training are, for example, Keras/Tensorflow, PyTorch, or . Eventually, the trained network needs to be deployed in a data

to be extended

processing hardware. Depending on the requirement of the data processing task, FPGAs, GPUs, or CPUs are utilized. The main parameters to decide which platform suits best to the application are the overall data bandwidth, the required latency, .

to be extended

The challenge of deploying machine learning in the processing of large-volume data streams is therefore both on the optimisation of the network architecture and on the hardware implementation. In particular for the latter, tools shall be further developed and provided to the research community, in order to allow higher-level optimisation of the data processing task.

add references

2.6 Frameworks for developments

add existing examples and a few concrete next steps

3 Connection to other WPs and TAs

3.1 Relation to TA2

Hermann

3.2 Relation to TA3

Dominik

Susanne

4 Use cases and workflows

4.1 Identification of pulsars/fast radio bursts

Ramesh

Identifying signals from short-lived (i.e $< 100\mu$s) radio bursts in streaming, high speed is a challenging problem, especially when dealing with multiple beamformed sky positions in parallel. For instance, the MeerKAT facility can generate > 500 beams at $\approx 50\mu$s time resolutions. Similarly when the SKA becomes operational, the facility will generate ~ 2200 beams. When a burst is detected, a trigger to capture the raw data stream (1.7 Tbits/s at MeerKAT) from the individual telescopes is generated. The detection of bursts is done by the real-time search on a sufficiently averaged, frequency-transformed beamformed-data over several dispersion trials. The latter is crucial as a significant dispersion in the signal ascertains its astrophysical origins.

Closely related, but more complex is the discovery of radio pulsars from the beamformed data from both the MeerKAT and the SKA. This problem is a truly multi-parametric search of the pulsar signal. As with radio bursts, the first step is to generate several time series each with different dispersion trial. Within a dispersed time series, the so-called acceleration and jerk trials are created (this process approximates the Doppler shift of the radio signals as they are emitted by a orbiting pulsar in a binary system, whose orbital parameters are not known in advance). Following this stage, a Fourier search for periodicity is carried out. Just as in radio burst searches, the pulsar search requires a suitably averaged and frequency transformed data, but lasting significantly longer in time. 30–minute

long data blocks are commonly used. However, before the next 30 minute-block arrives, the search over all beams, dispersion, acceleration and jerk trials should be complete. In contrast to the search for bursts, the pulsar searches do not require the raw telescope data streams.

4.2 Detection of anomalies in detector response

Arno - needs some re-writing

Anomaly detection is a method used for identifying rare or unusual signals in large data streams. Applications can be in physics measurements or in the operation and maintenance of physics instruments. In the latter case, it is also known as predictive maintenance, i.e. the detection of deviations from the usual behaviour well before a measurement instrument is in a critical mode of operation.

The result of the anomaly detection can be a tag of the physics event or a warning message indicating problems with the measurement device. These results will presumably become part of the regular data streams, like physics data or monitoring data.

Additional metadata may however be interesting to store. If an anomaly detection algorithm continuously updates its behaviour in order to learn from the regular data flow it will be useful to track the internal parameter settings of the algorithm. Another example are indicators which activated the anomaly detection. Specific signatures of the data or of the instrument behaviour may only be available in the real-time data stream and not in the data recorded during normal operation.

Different logging frequencies and metadata data volumes can thus be expected for anomaly detection. These should be evaluated in specific applications, so that general aspects can be identified and implemented in metadata concepts.

add existing examples and a few concrete next steps

4.3 Identification of particle decays in software-based triggering

Hans

Since 2022, the LHCb experiment uses a purely software-based high-level trigger (HLT) to decide which particle collisions provided by the Large Hadron Collider contain interesting processes. The HLT provides a full reconstruction of all signal candidates of interest online for every collision, which is made feasible with a large computing farm that uses CPUs and GPUs to process 40MHz of incoming data online. Only events which contain interesting process are stored, the rest is discarded. The ability of GPUs to run optimized algorithms massively in parallel was crucial to make the HLT technically feasible.

The HLT goes beyond the traditional trigger design, which consist of a fast hardware stage that uses low-level signals from the detector, and usually several successive software stages that perform a more and more sophisticated event reconstruction in each stage. Each stage discards a fraction of the events to reduce the processing rate for later stages. LHCb's HLT provides superior trigger efficiency compared to the traditional design, because there is only one stage, the

full information about the physical process contained in the event is available for the trigger to make the decision. In the traditional design, less information is available at trigger time and online decisions were less efficient. With the HLT, it is possible to make the optimal decision online for the first time.

Technically, a HLT adds high-level metadata to the raw recorded event, which is interpreted by an algorithm pipeline. At the lowest stage, tracks are build from hits in the tracking system and energy deposits in the calorimeters; particle identification data is attached. At the second stage, decay candidates are reconstructed from these tracks, according to rules which analysts have defined in advance. There are multiple decays of interest which are checked in parallel. For a typical two-body decay, the software forms all possible combinations of two tracks, and computes the mass of the ancestor under the hypothesis that the tracks are the products for decay of interest. If the mass of the candidate is close to the known mass of the parent particle, the trigger accepts the event. Further cuts, also defined in advance by the analysts, may be applied at this stage to the products of the decay to reduce combinatorial background.

4.4 Outside Triggers and sending Triggers

Jakob

5 Concept for dynamic life cycle of data

6 Conclusions

We have described concepts for identifying highly complex, multi-parametric signals arising in the context of irreversible data processing workflows. Our main focus is on data-intensive experiments within the PUNCH sciences, however most of the aspects in the concept could be generalized towards other fields of science also processing high data rates.

References

[1] The PUNCH4NFDI Consortium. Punch4nfdi consortium proposal, September 2020. This is the version documenting the work plan at the proposal stage. The reduction in funding led to a re-shaping of the work programme that is documented elsewhere.

[2] Qing Wu, Maksym Spiryagin, Colin Cole, and Tim McSweeney. Parallel computing in railway research. *International Journal of Rail Transportation*, 8(2):111–134, 2020.

[3] Xiaoli Cui, Pingfei Zhu, Xin Yang, Keqiu Li, and Changqing Ji. Optimized big data k-means clustering using mapreduce. *The Journal of Supercomputing*, 70(3):1249–1259, 2014.

[4] Chen Li, Yanfeng Zhang, Minghai Jiao, and Ge Yu. Mux-kmeans: Multiplex kmeans for clustering large-scale data set. In *Proceedings of the 5th ACM Workshop on Scientific Cloud Computing*, ScienceCloud '14, page 25–32, New York, NY, USA, 2014. Association for Computing Machinery.

[5] Rasool Azimi, Hedieh Sajedi, and Mohadeseh Ghayekhloo. A distributed data clustering algorithm in p2p networks. *Applied Soft Computing*, 51:147–167, 2017.

[6] Yaobin He, Haoyu Tan, Wuman Luo, Shengzhong Feng, and Jianping Fan. Mr-dbscan: a scalable mapreduce-based dbscan algorithm for heavily skewed data. *Frontiers of Computer Science*, 8(1):83–99, 2014.

[7] Irving Cordova and Teng-Sheng Moh. Dbscan on resilient distributed datasets. In *2015 International Conference on High Performance Computing & Simulation (HPCS)*, pages 531–540, 2015.

[8] Guangchun Luo, Xiaoyu Luo, Thomas Fairley Gooch, Ling Tian, and Ke Qin. A parallel dbscan algorithm based on spark. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, pages 548–553, 2016.

[9] Dianwei Han, Ankit Agrawal, Wei-Keng Liao, and Alok Choudhary. A novel scalable dbscan algorithm with spark. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1393–1402, 2016.

[10] Ilias K. Savvas and Dimitrios Tselios. Parallelizing dbscan algorithm using mpi. In *2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 77–82, 2016.

[11] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y. Chang. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):568–586, 2011.

[12] Zineb Dafir, Yasmine Lamari, and Said Chah Slaoui. A survey on parallel clustering algorithms for big data. *Artificial Intelligence Review*, 54:2411–2443, 2021.

[13] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, jan 2008.

[14] Matei Zaharia, Mosharaf Chowdhury, Michael Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 10:10–10, 07 2010.

[15] Deepa Kumar and M Rahman. Performance evaluation of apache spark vs mpi: A practical case study on twitter sentiment analysis. *Journal of Computer Science*, 13:781–794, 12 2017.

[16] Shen Huije and Tingwei Huang. *Spark for HPC: a comparison with MPI on compute-intensive applications using Monte Carlo method*. PhD thesis, 2018.

[17] José M. Abuín, Nuno Lopes, Luís Ferreira, Tomás F. Pena, and Bertil Schmidt. Big data in metagenomics: Apache spark vs mpi. *PLOS ONE*, 15(10):1–20, 10 2020.

[18] Manvi Saxena, Shweta Jha, Saba Khan, John Rodgers, Peggy Lindner, and Edgar Gabriel. Comparison of mpi and spark for data science applications. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 682–690, 2020.

[19] David Rohr. Data processing and online reconstruction, 2018.

[20] David Rohr, Sergey Gorbunov, and Volker Lindenstruth. Gpu-accelerated track reconstruction in the alice high level trigger. *Journal of Physics: Conference Series*, 898:032030, Oct 2017.

[21] Shreyasi Acharya et al. Real-time data processing in the ALICE High Level Trigger at the LHC. *Comput. Phys. Commun.*, 242:25–48, 2019.

[22] Ivan Kisel. Event topology reconstruction in the CBM experiment. *Journal of Physics: Conference Series*, 1070:012015, aug 2018.

[23] R. Aaij, J. Albrecht, M. Belous, P. Billoir, T. Boettcher, A. Brea Rodríguez, D. vom Bruch, D. H. Cámpora Pérez, A. Casais Vidal, D. C. Craik, P. Fernandez Declara, L. Funke, V. V. Gligorov, B. Jashal, N. Kazeev, D. Martínez Santos, F. Pisani, D. Pliushchenko, S. Popov, R. Quagliani, M. Rangel, F. Reiss, C. Sánchez Mayordomo, R. Schwemmer, M. Sokoloff, H. Stevens, A. Ustyuzhanin, X. Vilasís Cardona, and M. Williams. Allen: A high-level trigger on GPUs for LHCb. *Computing and Software for Big Science*, 4(1), apr 2020.

[24] Xiaocong Ai, Georgiana Mania, Heather M. Gray, Michael Kuhn, and Nicholas Styles. A gpu-based kalman filter for track fitting. 2021.