ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA151

FEASIBILITY OF HARDWARE ACCELERATION IN THE LHC ORBIT FEEDBACK CONTROLLER

L. Grech*, G. Valentino†, University of Malta, Msida, Malta
D. Alves, S. Jackson, J. Wenninger, CERN, Geneva, Switzerland

Abstract

Orbit correction in accelerators typically make use of a linear model of the machine, called the Response Matrix (RM), that relates local beam deflections to position changes. The RM is used to obtain a Pseudo-Inverse (PI), which is used in a feedback configuration, where positional errors from the reference orbit as measured by Beam Position Monitors (BPMs) are used to calculate the required change in the current flowing through the Closed Orbit Dipoles (CODs). The calculation of the PIs from the RMs is a crucial part in the LHC's Orbit Feedback Controller (OFC), however in the present implementation of the OFC this calculation is omitted as it takes too much time to calculate and thus is unsuitable in a real-time system. As a temporary solution the LHC operators pre-calculate the new PIs outside the OFC, and then manually upload them to the OFC in advance. In this paper we aim to find a solution to this computational bottleneck through hardware acceleration in order to act automatically and as quickly as possible to COD and/or BPM failures by re-calculating the PIs within the OFC. These results will eventually be used in the renovation of the OFC for the LHC's Run 3.

INTRODUCTION

Figure 1 illustrates the schematic of the Orbit Feedback Controller (OFC) as it is implemented today in the LHC. The red and blue arrows in Figure 1 show the data paths of the Beam Position Monitors (BPMs) and the Closed Orbit Dipoles' (CODs) signals respectively. The OFC uses BPM measurements throughout the machine in order to automatically adjust the average beam position by performing adequate changes to the COD currents [1].

The OFC was designed in C++ using ROOT libraries prior to the LHC start-up in 2008. ROOT is a scientific software framework originating from CERN with the purpose of being used for the analysis of experimental data related to high energy and nuclear physics [2]. The Service Unit (OFSU) was designed using CERN's Front-End Software Architecture (FESA) and serves as an interface to the OFC from which the operators can change certain parameters as well as control the operation of the OFC itself. The OFSU also serves as a proxy of the incoming measurements collected and calculations done by the OFC [3, 4].

The currents flowing in the CODs are related to the beam position as measured by the BPMs by a Response Matrix (RM), which essentially describes changes in positions as a function of COD deflections. The main principle behind the orbit correction is to (a) measure the beam position using the BPMs, (b) calculate the error with respect to the reference orbit and then (c) calculate the change in current needed in each COD to correct the beam position.

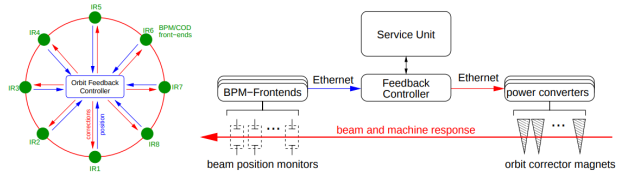


Figure 1: Schematic of orbit feedback controller [1].

To calculate the required change in current in each COD the OFC uses Singular Value Decomposition (SVD) algorithm, where the RM is decomposed into three more manageable matrices, from which the so called Pseudo-Inverse (PI) is found. This PI then directly relates the required COD deflections as a function of the measured orbit error [1].

The following is a basic description of the PI and the SVD algorithm. The PI of a non-square matrix A, is a matrix which when multiplied to A produces an identity matrix. The PI is calculated after computing the SVD of A, which evaluates A as a multiplication of three sub-matrices with special properties, making it trivial to find the PI. Below is the result after SVD:

$$A = U\Sigma V^*$$

where U and V are unitary matrices and Σ is a rectangular diagonal matrix, with non-negative real numbers on its diagonal. The PI, A^{-1}, is calculated as follows:

$$A^{-1} = V\Sigma^+ U^*$$

where Σ^+ is Σ with the diagonals inverted, i.e. $\Sigma^+_{ii} = \frac{1}{\Sigma_{ii}}$

In the OFC, the SVD is used to compute the PIs of the horizontal and vertical RMs, which have around 1150 rows (number of BPMs) and 550 columns (number of CODs in one plane). Apart from this there might be different configurations of the LHC (optics) in which the magnets are used, for which RMs and accompanying PIs have to be computed as well.

PROBLEM DEFINITION

In the current implementation of the OFC, multiple response matrices (e.g. during the RAMP, 13-14 RMs are used) for different machine configurations (optics) are used. It might be the case that there is a change in the LHC operation due to a malfunction in the BPMs or the CODs and

* leander.grech.14@um.edu.mt
† gianluca.valentino@um.edu.mt

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA151

this would prompt a change in the RM as well. Considering that it takes in the order of seconds to re-calculate a PI from a RM using SVD, the OFC is unable to react quickly to changing LHC parameters and therefore this functionality is suppressed in the OFC. As a workaround when the LHC operators suspect a malfunctioning COD which is causing a residual error in the orbit correction, it is masked and thus current requests are no longer sent to it. Following this the PI is calculated offline on another machine using SVD and then is manually uploaded to the OFC where it can be used without the malfunctioning COD. This has been shown to work in the past with non-critical CODs however it is not a sustainable solution mainly because the whole process can take from a few minutes to a few a hours to complete, depending on the availability of expert operators who can perform this task.

In the case of a BPM suspected to be malfunctioning, the LHC operators can mask the respective BPM so that its measurements are no longer taken into consideration during the orbit correction calculations. This may also be done automatically by the OFC itself and is accomplished by reading the hardware state given to it by the BPM UDP packets coming from the respective BPM crates. Ideally, following the masking of a BPM, the PI should be recomputed immediately, but due to the long computation time, the reading of that BPM is instead set to the reference value.

Ultimately in the current OFC implementation, the required COD deflections are calculated using the original PI without taking into consideration the malfunctioning BPM. The result of this is that the beam position at the location of the malfunctioning BPM is left relatively unaffected by artificially forcing the position at that location to be exactly the reference orbit [4]. This is an adequate solution when a small number of BPMs are malfunctioning in sparse locations, however should it be the case that adjacent BPMs are defective, the OFC might induce significant beam distortions. Thus it would be a significant improvement over the current system to calculate the PI immediately and in real-time for any malfunction in the BPMs.

As opposed to the solution mentioned above for the BPMS, the situation for malfunctioning CODs is more critical. When we multiply our positional errors vector with the PI calculated by the SVD, we are using the entire set of corrector magnets included in the RM. Applying a correction involving a malfunctioning corrector may lead to important distortions not only around a failing corrector but also throughout the whole machine. The orbit will reflect the absence of the corrector kick and the PI will automatically try to restore the missing kick. As a consequence, in the event of a corrector failure the feedback must be stopped and the SVD on all response matrices recomputed without that corrector.

There are 2 categories of correctors: (1) $\approx$ 250 correctors with maximum currents between 70A and 600A where in case of a powering failure an automatic dump trigger is generated and the beams are dumped immediately. (2) 756 arc correctors (CODs) with a maximum current of 60A where in case of a failure a dump trigger is generated if the corrector deflection is larger than 5 µrad.

Since the RMS kick strength of the CODs is around 12 µrad, the beam is dumped for the majority of corrector failures (5-10 events per year), however for smaller deflections the OFC can correct the problem even if the PI is not recalculated immediately. This is due to the circuit time constants of these arc corrector magnets which range between 20 and 60 seconds, thus creating a sufficiently long "beam rescue window". These failures have to be detected immediately by the OFC by reading the state information of the power converters powering the CODs and then updating the PI matrices by excluding the malfunctioning corrector within no more than 2 seconds. The missing corrector will leave a bump on the orbit but using this process it is possible to continue operation in these conditions and thus saving the beam.

The crux of this work was the recalculation of the new PI matrix through the SVD algorithm as it has been found to be the computational bottleneck of the OFC in this regard. Regardless of whether the component failure comes from the CODs or the BPMs, the aforementioned PI has to be recalculated in order to ensure the most optimal beam control and consequently maximising the beam lifetime. Since the SVD algorithm is very computationally expensive for relatively large matrices ($O\left(mn^2\right)$, where m is the number of BPMs and n is the number of CODs), care should be taken as not to compromise the the real-time behaviour of the OFC.

PROPOSED SOLUTION

For the case when the COD is not in a critical location in the LHC, we propose that one of the heaviest computational components in the OFC today, which is the pseudo-inverse calculation through the SVD algorithm, could be made faster by using hardware acceleration. Hardware acceleration libraries allow the use of multi-core devices, such as modern-day conventional CPUs, much more efficiently through a careful use of the CPU cache memory. Apart from this, such libraries allow you to utilise other devices such as Graphical Processing Units (GPUs) which have a massive number of computational units.

Today there exist many GPU manufacturers, with the main two brands being NVIDIA and AMD, however NVIDIA is often considered at the forefront of the competition. NVIDIA also developed a parallel computing platform, CUDA, for the sole purpose of creating programs which can use both the CPU and GPU for mathematical calculations [5]. There exists another parallel computing platform, OpenCL, which unlike CUDA is non-proprietary and royalty-free [6]. For the tests described herein, only NVIDIA devices were considered, namely the GTX1080Ti and the Tesla V100, primarily due to their availability as offered by CERN's Techlab [7].

After some research we found a hardware acceleration library called ArrayFire which might be ideal for the calculation of the SVD algorithm within the OFC [8]. ArrayFire abstracts the hardware drivers from the programmer and pro-

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA151

vides an easy-to-use interface with which to accelerate code. Apart from this ArrayFire provides a myriad of hardware accelerated mathematical functions ranging from simple array manipulations to multi-dimensional matrix manipulation and linear algebra. ArrayFire also provides the user with a choice as to which device, as well as which programming paradigm to use for code acceleration.

In particular ArrayFire gives the programmer the ability to choose from (1) OpenCL, (2) CUDA and (3) asynchronous multi-threaded execution on multi-core CPUs. Both (1) OpenCL and (2) CUDA offer the possibility to interface with massively parallel architectures including, but not only limited to GPUs whereas (3) the CPU implementation uses optimised thread manipulations depending on the required implementation.

TESTS

All the following tests were done on different devices, each having different kinds of hardware. Specifically these devices consisted of multi-core CPUs, a high-end gaming oriented GPU (GTX1080Ti) and a high-end general-purpose GPU (Tesla V100).

Benchmarking tests on each device were done, where the ArrayFire and the original implementation using ROOT objects of the SVD algorithm were tested. Specifically these tests focused on measuring the time it takes the device to compute the PI of the response matrix, and finding the average execution time over many executions. Following this another benchmarking test was done on the ArrayFire implementation focusing on collecting more statistics on the execution time of the PI calculation.

Another test focused on comparing the PI provided by the Orbit Feedback Service Unit (OFSU) with that calculated by using ROOT objects and the ArrayFire libraries on the different machines available. In this test the PI matrix was multiplied with the original response matrix, which obtained a pseudo-identity matrix. The latter was then compared to an identity matrix from which some accuracy statistics were obtained. Such statistics include the average error from the ideal identity matrix as well as the standard deviation of the errors.

The final test done focused on measuring the machines' resource usage. This was done by running the benchmarking test with one iteration of the PI calculation and measuring the CPU usage as well as the memory consumption of the process. Following this a python script was used to visualise the measured data from the running process.

RESULTS

Benchmarking Tests

The benchmarking tests provided a good insight into the effect of hardware accelerated libraries on the computation time of the SVD algorithm. Figure 2 shows the average execution time of the SVD algorithm for different libraries, running on different machines. Note that some libraries were not possible to be used on specific machines, e.g. the ArrayFire CUDA library could not be used where there was no NVIDIA device. The ROOT and ArrayFire CPU libraries were executed on all machines and for those machines which have a GPU installed, these libraries were executed on their respective CPUs.

From the results shown on Figure 2, it is clear that the fastest libraries were the ArrayFire CPU and ArrayFire OpenCL which on average outperformed the traditional ROOT implementation by two orders of magnitude. The execution time on Intel Core i7-4790 was similar for both the ArrayFire CPU and OpenCL implementations however the small difference between the two execution times could not attest which one of the two libraries performs better. The execution times for both Intel Xeon Co-processors were also very similar to one another, however it is clear that the Xeon Phi 7120 outperformed the Xeon E5-4650 on all occasions. From the results below it can also be concluded that the ArrayFire CUDA implementation on both NVIDIA devices was on average, approximately half an order of magnitude slower when compared to both the ArrayFire CPU and OpenCL implementations.

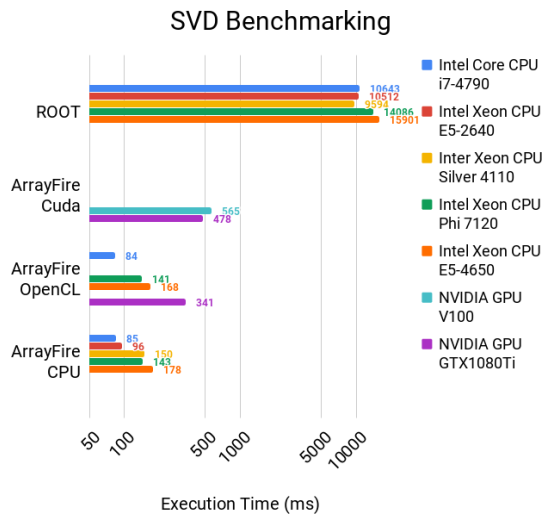


Figure 2: SVD execution time on different devices.

Another important conclusion drawn from this test was that the use of GPUs, namely the NVIDIA GTX1080Ti and the NVIDIA V100, is not suitable for the SVD calculation of the PI of the LHC response matrix as all of the CPU machines outperformed them. The reason for the poor performance of the GPUs in this implementation is that the speedup obtained from a massively parallel architecture of a GPU is overshadowed by the time it takes to transfer data, to and from such devices. Should it have been the case where the SVD had to be performed on a much larger matrix, such as a high-definition image, the use of GPUs would have proved to be ideal. It is important to note that GPUs are mounted on the motherboard via a PCI (Peripheral Component Interconnect) port, which is inherently slower than the connection between the CPU and main memory. In addition, ArrayFire optimises the SVD algorithm through the use of

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA151

the CPU cache, which is orders of magnitude faster than main memory [9].

SVD Accuracy Statistics

This test addresses the hypothesis that different device architectures (namely in GPUs) would provide different results due to different optimisation techniques when hardware accelerating the SVD algorithm. The different results might be caused by floating point rounding errors or different cut-off values used within the SVD algorithms themselves as they are implemented in the respective libraries. Note that all the results calculated by all the libraries were compared to the original PI of the respective RM which has been used in the OFC. Throughout the rest of this test these results will be annotated by OFSU (Orbit Feedback Service Unit) results, as they were obtained from this FESA class.

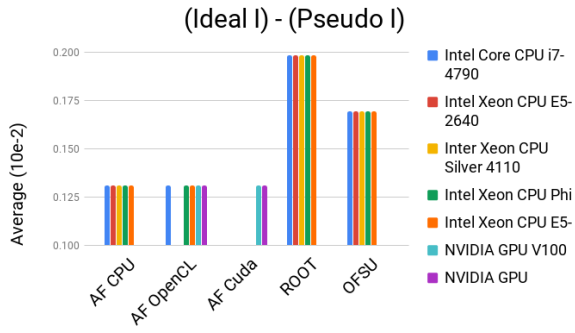
To obtain results for this test first a pseudo-identity matrix was obtained by multiplying the original RM by the PI obtained from the respective computation library calculated on each respective machine as shown in Figure 2. Following this an absolute error of all the elements of the pseudo-identity matrix was obtained by subtracting an ideal identity matrix from the pseudo-identity matrix. The average error and the standard deviation of this absolute error from an ideal matrix were then used as metrics for the accuracy of the respective method by which the PI was calculated.

Figure 3a shows that the average error obtained from respective libraries is the same when executed on different machines. This suggests that the computation result of the PI through the SVD algorithm is not hardware dependent. Figure 3a also shows a discrepancy in the error obtained from the OFSU and the ROOT libraries, however this should not be taken into consideration in this test. This discrepancy was due to calculation parameters which could not be perfectly matched from one library to the other. Such parameters include the number of eigenvalues (singular values) to be used in the calculation of the PI as well as the cut-off value for such eigenvalues, below which the eigenvalue is zeroed out.

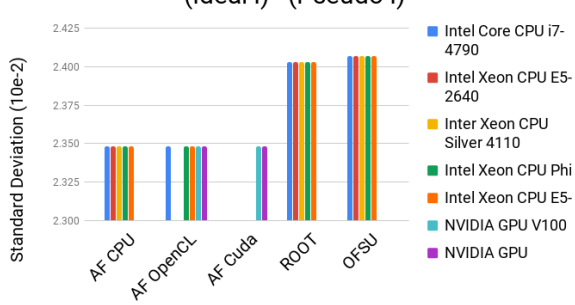
The standard deviation of the error is shown on Figure 3b where it can be seen that this value is independent of the hardware where the computation is running on, as was the case for the average error. It can also be seen that the standard deviation of the error is approximately the same for all the libraries with which the PI is calculated. This implies that the accuracy of the calculations is similar across libraries.

Memory and CPU Usage

The best performing device from the above tests was the Intel® Core™ i7-4790 CPU @ 3.60GHz CPU. Following this, CPU and memory usage analysis were done on all three libraries supported by this device. First off was the implementation using the ROOT libraries, and it can be seen from top plot in Figure 4 that even if multiple cores are being used throughout the execution of the program, they are not



(a) Average



(b) Standard deviation

Figure 3: Error of pseudo-identity matrix from an ideal matrix.

being used simultaneously at any point in the execution of the program.

This is different to when the ArrayFire libraries are used. The bottom plot in each of Figure 4 and Figure 5, respectively show the ArrayFire CPU implementation with multiple threads and here it can be seen that the approximate time that these multiple threads stay *alive*, is in accordance to the results obtained in Figure 2, which was ≈ 80 ms. Similarly for the program when it is built with OpenCL libraries, it can be seen from the middle plot in Figure 4 and the top plot in Figure 5 that for ≈ 80 ms all of the CPU cores were used.

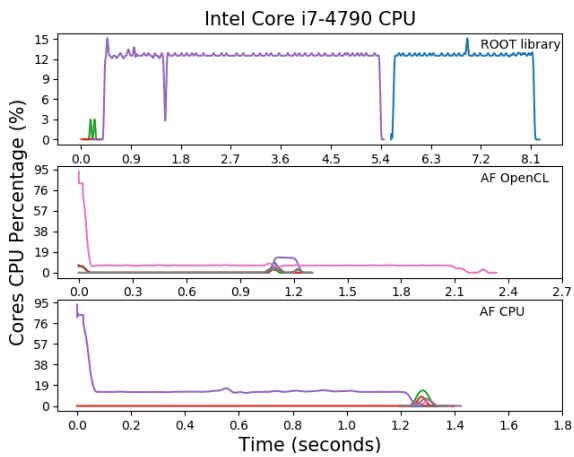


Figure 4: CPU usage percentage during SVD calculation using different libraries.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOPHA151

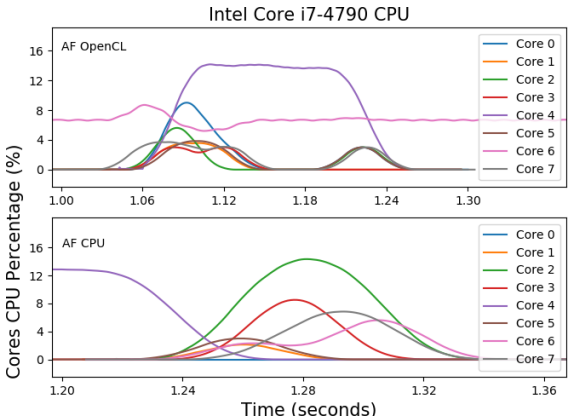


Figure 5: Details from Figure 4.

Apart from the CPU utilisation, the machine's main memory (RAM) usage was also analysed. Here Figure 6 shows the memory usage for the original SVD algorithm using ROOT objects; ArrayFire using CPUs; and ArrayFire using OpenCL implementations respectively. The original implementation is quite standard, where after approximately 2 seconds, the response matrix is loaded and enough memory is allocated to hold the PI after it is calculated. This memory consumption remains constant up until the end of the execution.

For ArrayFire over CPU in Figure 6 (green plot) we see a different scenario, where after 1.2 seconds, the memory consumption increases by approximately one order of magnitude. This is due to ArrayFire copying the input response matrix over many threads, in order to parallelize the algorithms used to calculate the PI. A similar situation can be seen for ArrayFire over OpenCL with the difference being that this implementation uses three times more memory during the calculation of the PI.

This difference in memory consumption between ArrayFire's CPU and OpenCL libraries can be attested to the specific programming paradigm used when opting to use OpenCL. However the latter was not further analysed due to it being proven to have a similar performance to the ArrayFire CPU implementation yet consuming close to three times more memory.

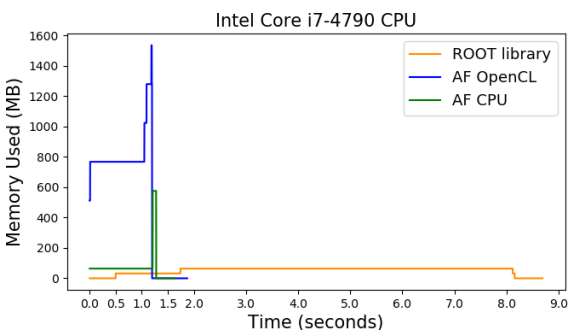


Figure 6: Memory consumption during SVD calculation using different libraries.

CONCLUSION

From the work presented in this paper, it was concluded that hardware acceleration libraries offer a good solution to solve the computational bottleneck found in the current implementation of the OFC when it is calculating the pseudo-inverse for the response matrix used between the CODs and the BPMs. However the use of GPUs in future designs of the OFC would not be recommended due to the relatively small matrices used in the calculations. It was found that the time overhead of copying the input matrix to the GPU would balance out the accelerated computation by the GPU. In conclusion the preferred choice for accelerating the SVD computation within the OFC is ArrayFire using CPUs.

REFERENCES

[1] R. J. Steinhagen, "LHC beam stability and feedback control-orbit and energy", PhD thesis, RWTH Aachen U., 2007.

[2] ROOT a data analysis framework, https://root.cern.ch/. Accessed:2018-8-15.

[3] L. Fernandez *et al.*, "Front-end software architecture", in *Proc. 11th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'07)*, Oak Ridge, TN, USA, Oct. 2007, paper WOPA04, pp. 310–312.

[4] J. Wenninger and R. Steinhagen, "LHC orbit feedback control requirements", CERN, Geneva, Switzerland, Technical Rep. CERN AB-OP, March 2007.

[5] CUDA (compute unified device architecture) definition, July 2015, https://techterms.com/definition/cuda. Accessed: 2018-12-3.

[6] OpenCL - the open standard for parallel programming of heterogeneous systems, July 2013, https://www.khronos.org/opencl/. Accessed: 2018-12-3.

[7] Techlab, https://techlab.web.cern.ch/. Accessed: 2018-12-3.

[8] J. G. Malcolm *et al.*, "ArrayFire: a GPU acceleration platform", in *Proc. SPIE, Modeling and Simulation for Defense Systems and Applications VII*, vol. 8043, pp. 49–56, May 2012. doi:10.1117/12.921122

[9] D. Levinthal, "Performance analysis guide for intel® core i7 processor and intel® xeon 5500 processors", Intel Corporation, Technical Report, 2009.