Near Memory Acceleration on High Resolution Radio Astronomy Imaging

Stefano Corda[1,2], Bram Veenboer[3], Ahsan Javed Awan[4], Akash Kumar[5], Roel Jordans[1], Henk Corporaal[1]

[1]Eindhoven University of Technology [2]IBM Research Zurich [3]Astron [4]Ericsson Research [5]TU Dresden

{s.corda, r.jordans, h.corporaal}@tue.nl, veenboer@astron.nl, ahsan.javed.awan@ericsson.com, akash.kumar@tu-dresden.de

Abstract—Modern radio telescopes like the Square Kilometer Array (SKA) will need to process in real-time exabytes of radio-astronomical signals to construct a high-resolution map of the sky. Near-Memory Computing (NMC) could alleviate the performance bottlenecks due to frequent memory accesses in a state-of-the-art radio-astronomy imaging algorithm. In this paper, we show that a sub-module performing a two-dimensional fast Fourier transform (2D FFT) is memory bound using CPI breakdown analysis on IBM Power9. Then, we present an NMC approach on FPGA for 2D FFT that outperforms a CPU by up to a factor of 120x and performs comparably to a high-end GPU, while using less bandwidth and memory.

I. INTRODUCTION

The first phase of the Square Kilometre Array (SKA), the biggest radio-telescope project in the world, has very high-performance requirements, more precisely current estimates range in the order of Exaflops and PetaBytes per second [1]. Especially, high-resolution image processing is crucial to detect less bright and distant sources. In particular, Image Domain Gridding (IDG) [2], the state-of-the-art radio-astronomy gridding, and degridding algorithm, is a novel method employed in radio-astronomical imaging. It consists of different algorithmic steps (*Fig. 3*), which are differently affected by image resolution. As shown in *Fig. 1*, while some kernels, Gridder and Degridder, perform quite well even increasing the image size, we observe that the 2D FFTs exponentially become the application bottleneck since it is memory bound, and it does not reach peak performance. *Fig. 1* shows the contribution of FFT in the execution time of IDG at different image sizes.
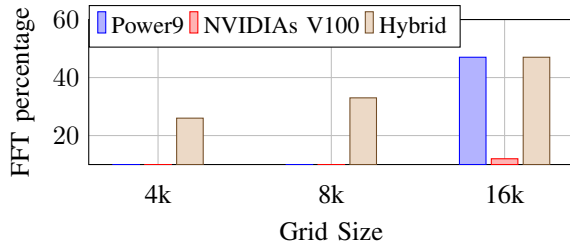

Fig. 1: *Percentage of the IDG execution time spent on 2D FFT. The maximum (100%) is the total execution time of IDG. The Hybrid system consists of running the FFT on the CPU and the Gridder/Degridder on GPU. This solution has been adopted because for larger image size (e.g. 32k points per dimension) the GPU does not have sufficient memory.*

While the prior art has focused on accelerating gridding and degridding functions in the IDG pipeline using GPU [2] and FPGA [3], we focus on accelerating the FFT function.

By employing hardware performance counters on IBM Power9, we aim to identify performance bottlenecks in the high-resolution radio-astronomy gridding and degridding imaging application. Especially memory bottlenecks could be critical in this domain like in other big-data applications [4] due to the inefficient use of on-chip cache hierarchy, memory wall [5] and the end of Dennard scaling [6]. Near-Memory Computing (NMC) [7]–[9] tries to overcome these limitations by moving the processing to where the data is located, as opposed to the classical compute-centric approach of moving the data through the entire cache hierarchy. Furthermore, NMC employs new 3D-stacked memory technologies such as HBM2 in this work, which has high off-chip memory bandwidth. We evaluate the efficacy of the NMC approach for 2D FFT acceleration. Our key contributions are:

- We apply a CPI breakdown analysis to the state-of-the-art radio-astronomy algorithm to detect memory bottlenecks and, we select the performance monitoring units (PMUs) on IBM Power9 that help in identifying memory boundness, in this case, represented by 2D FFTs. Furthermore, we observe in the radio-astronomy imaging context how these counters vary with the increasing image resolution and, consequently, the memory boundness.
- We compare three different architectures showing how an NMC platform on FPGA can alleviate the memory bottleneck caused by large 2D FFTs outperforming IBM Power9 and achieving performance comparable to GPU using less memory and bandwidth.

This paper is structured as follows: *Section II* presents the essential concepts on radio-astronomy and hardware performance counters. In *Section III* we explain our methodology. Then, *Section IV* shows the application characterization analysis and the system evaluation. Related works are discussed in *Section V* and *Section VI* concludes the paper.

II. BACKGROUND

This section presents the radio-astronomy imaging (*II-A*) and the CPI Breakdown analysis (*II-B*) background.

A. Radio-Astronomy Imaging

One of the main challenges in radio-astronomy is to translate the incoming signals from the sky to a sky image (*Fig.*

2). This process consists of the following steps: ❶ digitization of the incoming electromagnetic waves from radio sources in the universe; ❷ correlation of the digitized signals produced by pairs of distinct stations, which produces the measurement data (`visibilities`); ❸ the calibration step estimates and corrects instrument parameters and environmental effects; the partially corrected visibilities are converted into a sky image by an imaging step ❹.
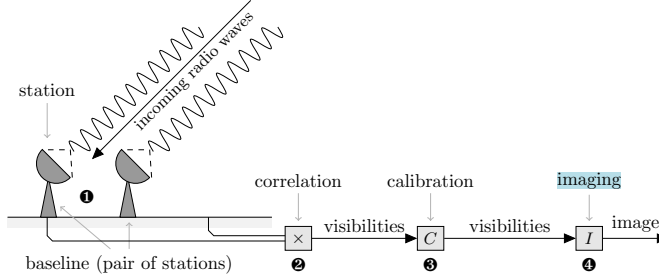

Fig. 2: *Radio astronomy image acquisition [2].*

This paper focuses on step ❹, in particular on the current state-of-the-art gridding and degridding algorithm (*Fig. 3*) called Image Domain Gridding (IDG) [2]. The imaging step starts with an empty sky model and it consists of an iterative 3-steps process: ① the `imaging` step is performed on the measured visibilities producing the `residual image`; ② a variant of the CLEAN is employed to extract one or more bright sources, which masks the more interesting weak sources, and is added to the `sky model`; ③ the visibilities of the extracted sources are `predicted` and then subtracted to the input to reveal fainter sources.
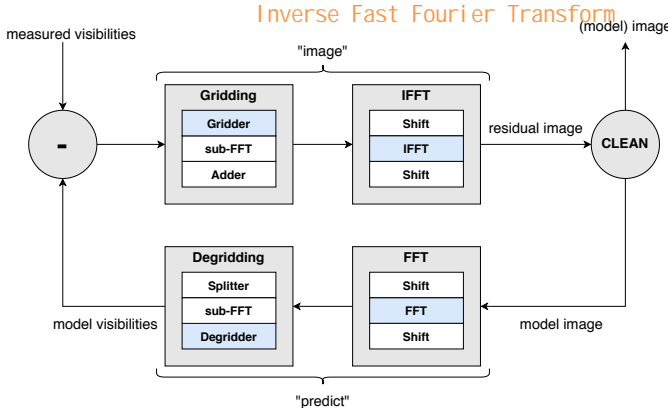
Fig. 3: *Complete Radio Astronomy Imaging Step [2]. Image Domain Gridding is a highly efficient implementation of the gridding and degridding steps (image and predict), while it leaves the CLEAN execution to other imaging application such as WSCLEAN [10].*

Furthermore, we show in *Fig. 3* that all these kernels consist of sub-kernels, of which the light-blue ones (Gridder, Degridder, FFT) are the ones executing most of the time (over 95% on average for the considered image resolutions), thus focusing our work on them.
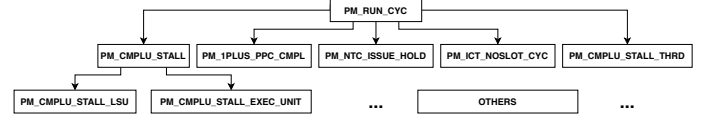
B. CPI Breakdown Analysis

Fig. 4: *Power9 CPI Breakdown tree [11].*

While Intel architectures can be studied employing approaches/tools such as Top-Down [12] or Intel VTune [13], IBM Power architecture lacks in this space. In this work, we focus on the IBM Power9, which can be analyzed using the same methodology presented in [11] for IBM Power8.

TABLE I: *IBM Power9's Performance Monitoring Units (PMUs) description.*

PMU	Description
PM_RUN_CYC	Run cycles
PM_CMPLU_STALL	Nothing completed and ICT is not empty
PM_CMPLU_STALL_THRD	Completion stalled because the thread was blocked
PM_1PLUS_PPC_CMPL	One or more PPC instructions finished
PM_NTC_ISSUE_HOLD	NTC instruction is held in the issue
PM_ICT_NOSLOT_CYC	Number of cycles the ICT has no itags assigned to this thread
PM_CMPLU_STALL_LSU	Completion stalled by an LSU instruction
PM_CMPLU_STALL_EXEC_UNIT	Completion stall due to execution units (FXU/VSU/CRU)

PMUs are programmable components contained inside each microprocessor core on the chip. They are used to collect and filter information gathered from various aspects of the chip and they can attribute the events to the threads within the core. Power9 supports around 1000 PMU[1] events that can be monitored. The CPI Breakdown consists of creating a breakdown of the total run cycles in different categories, e.g. stalls in load/store units, to understand where the application is spending most of the time, thus being able to detect application bottlenecks. A simplified representation, containing the most interesting PMUs for memory bottlenecks (see *Section* III), of the CPI breakdown is reported in *Fig. 4* and relative meaning in *Tab. I*.

TABLE II: *System parameters and configuration.*

IBM Power9 AC922 CPU	@3.8 GHz, 22 cores (4-way SMT), 2 sockets, 32 KB L1 cache per core, 256 KB L2 cache per core, 120 MB L3 cache per chip, 512 GB DDR4 2666 MHz
NVIDIA V100-SXM2-32G GPU	@1.53 GHz, 640 Tensor Cores, 5120 NVIDIA CUDA Cores, NVlink interconnect 300 GB/s 32 GB HBM2 at 900 GB/s
AlphaData 9V3	788 FFs, 394k LUTs, 2280 DSPs, 25.3 Mb BRAM, 90.0 Mb URAM, 8 GB DDR4 2400 MHz
AlphaData 9H7	2607 k FFs, 1304 k LUTs, 9024 DSPs, 70.9 Mb BRAM, 270 Mb URAM, 8 GB HBM2 at 460 GB/s

III. METHODOLOGY

In this section we show the system (*III-A*) and the tools/software (*III-B*) we use for our work.

---

[1]https://wiki.raptorcs.com/w/images/6/6b/POWER9_PMU_UG_v12_28NOV2018_pub.pdf

A. System in use

In *Fig. 5*, we present the system employed for this work. We employed an IBM Power9 AC992 with 22-cores SMT4, more details are in *Tab. II*. We include as a competitor to NMC an NVIDIA V100, one of the latest GPU with 32GB of HBM2 memory at $900\,\text{GB/s}$, which uses similar technology to the NMC platform. As NMC systems we use a custom hardware design called `Access Processor (AP)` [14], which can be mapped on different FPGAs (DDR4 and HBM2).
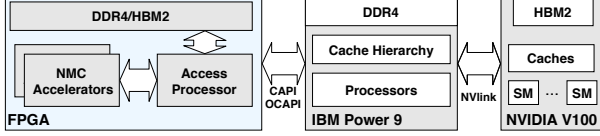

Fig. 5: *System employed [14].*

Double Data Rate:scalable

High Bandwidth Memory:Higher bandwidth, lower power consumption, higher performance

Differently from a classical general-purpose computer, where the access bandwidth and latency depend on a complex mixture of workload characteristics and the memory hierarchy, the `Access Processor (AP)` design comprises the so-called memory controller, which has the feature of enabling more control over the memory system and programming all the concurrently running data streams from/to the attached NMC accelerators (see *Fig 5*). The key features of the `AP` are: 1) the B-FSM, a programmable state machine technology, applied successfully to a wide range of co-processor devices [15]; 2) programmable address mapping scheme that can highly optimize the bandwidth utilization reducing bank conflicts and managing the data organization. 2D FFT acceleration on `AP` is performed as a combination of multiple 1D FFTs and transpose (see *Fig. 6*). It consists of performing a 1D FFT over all the rows of the image and an on-the-fly partial matrix transpose. Then, a 1D FFT is performed on the transposed columns of the images and they are transposed again on-the-fly. In this work we employed performance estimation, which is conservative, for the `AP` based on experiments, e.g running 1D FFTs and matrix transpose.
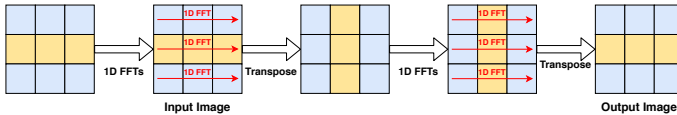

Fig. 6: *2D FFT decomposition in 1D FFTs and matrix transpositions.*

B. Tools and Software

As a small experiment, testing our tools and analysis methodology, we show in *Fig. 7* the CPI breakdown analysis (y-axis shows the PMU percentage over the total run cycles) applied to three simple benchmarks: mac, which is a compute-bound kernel written using IBM Power9 intrinsics that perform fused multiplication and accumulate over the same array of data; sgemm, which is a single-precision general matrix to

matrix multiplication; stream-add, a common memory-bound benchmark used to compute the peak bandwidth of a system. We make the following two observations. First, the PMUs not included in the bar chart are nearly 0%, thus explaining the relevance of the selected counters. Second, we can distinguish clearly a separation between a kernel completely memory bound (stream-add), which spends most of the time stalling on LSU (load-store units), and another one compute-bound (mac), which spends most of the time stalling on the computing units.
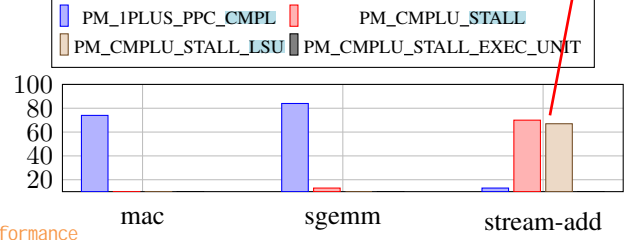

Fig. 7: *CPI Breakdown applied to test cases.*

FFT was run on CPU using FFTW3 version 3.3.8 and on GPU using cuFFT of the CUDA library version 10.1. Furthermore, we improved the Degridder and Gridder algorithms on Power9 porting the Intel-based code employing IBM Power9 intrinsics. In particular, the main optimization was to use a sine/cosine lookup table, which was implemented with AltiVec intrinsics[2]. Especially this section of the algorithm is challenging on other CPU platforms as well; for instance on Intel high performance is obtained employing MKL (math kernel library), which is not available on PowerPC. We use IDG[3] version `5736086c` employing the parameters in *Table III*.

case configuration

TABLE III: *Image Domain Gridding parameters.*

Parameters	Values
Stations	120
Channels	16-32
Timesteps	8192
Grid Size	4096-8192-16384
Sub-grid Size	32
Cycles	1
Grid Padding	1.0

In order to characterize the application we employed `perf` [16] for extracting the PMUs values.

IV. RESULTS

We discuss the application characterization results in *IV-A* and the evaluation of three platforms for the 2D FFT kernel in *IV-B*.

A. Application Characterization

We present the CPI breakdown analysis applied to Image Domain Gridding on IBM Power9 in *Fig. 8*. More precisely, we show the trend of the most interesting performance counters (y-axis shows the PMU percentage over the total run

---

[2]https://openpowerfoundation.org/?resource_lib=power-isa-version-3-0
[3]https://gitlab.com/astron-idg/idg
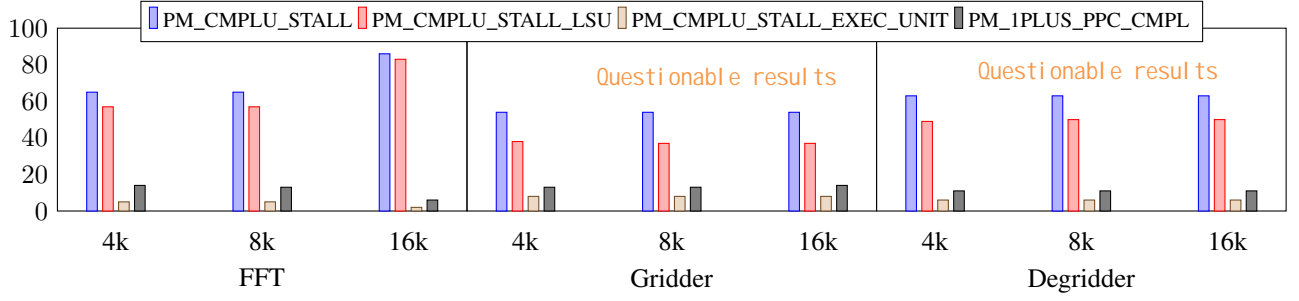
Fig. 8: *Power9 CPI Breakdown analysis of Image Domain Gridding.*

(a) *Image Domain Gridding on IBM Power9.*

(b) *Image Domain Gridding on NVIDIA V100.*

(c) *2D FFT on* `Access Processor` *with DDR4.*

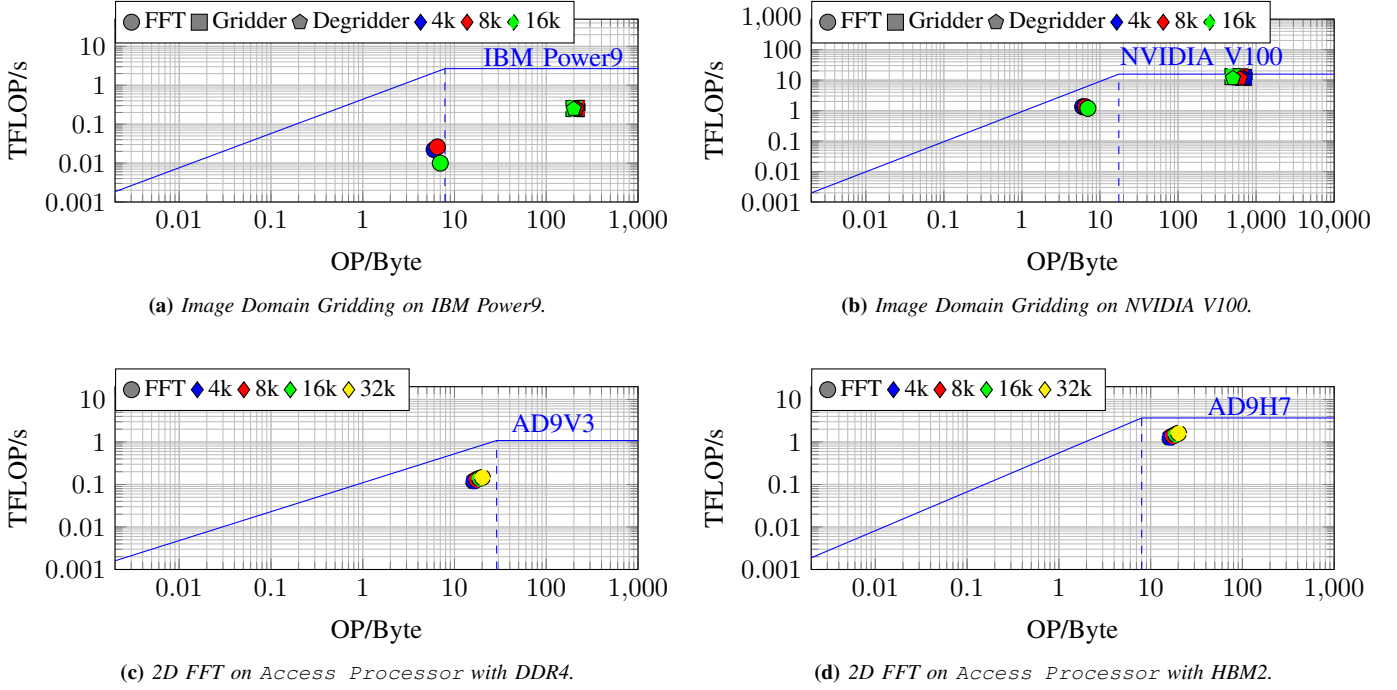**(d)** *2D FFT on* `Access Processor` *with HBM2.*

Fig. 9: *Roofline Analysis of Image Domain Gridding.*

cycles) on increasing the visibilities grid size. The FFT spends more time on stalling on the load-store units compared to Gridder and Degridder, which means it is more memory bounded. Moreover, FFT spends less time on stalling on the execution units. Furthermore, the FFT becomes increasingly memory bound with larger grid sizes (see 16k vs 8k in *Fig. 8*) reflecting in a larger time spent on executing it (see *Fig. 1*).

We further analyze the application on IBM Power9 employing the well-known technique of the roofline model [17]. Power9's bandwidth is $340\,\mathrm{GB/s}$ for 2 sockets and the peak performance is estimated employing the following formula:

$$TFlops = \frac{freq\,[GHz] * \#_{op.\,per\,core} * \#_{cores} * \#_{sockets}}{1000}$$

Each core of the IBM Power9 can perform 16 parallel single precision operation. Using the other information from *Tab. II* we get $2.675\,\mathrm{TFlops}$.

Fig. 9a shows the roofline model for the kernels in Image Domain Gridding. In particular, we notice that FFT is memory bounded as it is underneath the peak bandwidth ceiling while Gridder and Degridder are compute-bound since they are underneath the peak performance ceiling. Furthermore, the FFT with a grid-size of 16k shows lower performance compared to the FFT performed with smaller grid-sizes. This behavior is due to the larger amount of time spent on stalling in the LSUs. Furthermore, the performance on Power9 remains low compared to the other architecture.

We also include the roofline model of Image Domain Gridding on NVIDIA V100 (see *Fig. 9b*). Peak performance is reported on the card datasheet ($900\,\mathrm{GB/s}$ and $15.7\,\mathrm{TFlops}$). On NVIDIA V100 IDG achieves higher performance compared to Power9 for similar kernel characteristics. Furthermore, we build the roofline model for the 2D FFT on `Access`

`Processor` employing the methodology proposed by Intel[4]. More precisely, using this methodology, which estimates the peak performance computing the maximum number of adders that can fit on the FPGA consuming all the DSPs and the logic cells, we compute the peak performance for the two FPGA boards respectively of $1.080\,\mathrm{TFlops}$ and $3.675\,\mathrm{TFlops}$. The maximum memory bandwidth is $37.5\,\mathrm{GB/s}$ for 2 DDR4 bank at $2400\,\mathrm{MHz}$ and $460\,\mathrm{GB/s}$ for the HBM2. We show that using the FPGA with DDR4 the FFT reaches higher performance compared to Power9 and it is memory bound (see *Fig. 9c*). Contrariwise, the higher bandwidth on the FPGA with HBM2 further increases the performance and makes the kernel compute-bound (see *Fig. 9d*). Moreover, the FPGA has similar performance compared to GPU having lower peak bandwidth and peak performance. The more efficient use of the memory is shown in *Fig. 9a*, where the arithmetic intensity achieved by the FPGA is higher.

B. Offloading on NMC Systems

The `AP` provides fine-grained control to schedule the accesses to the DDR4 and HBM2 memory (see *Fig. 10*), the transfer of the data to and from the FPGAs internal SRAM (Block RAM and/or UltraRAM), and the processing of the data [14]. Because the various 1D FFTs (see *Fig. 10*) are calculated in parallel using multiple accelerators (the 1D FFTs design used is taken from [18]), the `AP` can schedule the transfer of the input data for each 1D FFT computation from a DDR4 DIMM or HBM2 memory channel to a given accelerator during the time that additional 1D FFTs are being computed on the other accelerators. The same applies to the transfer of the 1D FFT results from an accelerator back to the DDR4 or HBM2 memory. As a result, the access, transfer, and processing of the input data and results for the 1D FFT calculations on the rows of the matrix can be overlapped in an almost seamless fashion, which enables to obtain very high performance by achieving near-optimal utilization of the available DDR4 or HBM2 memory bandwidth [19]. In this case, the 2D FFT performance will be determined almost entirely by the available memory bandwidth, on the condition that there are enough accelerators available to fully overlap the memory access and transfer times. Experiments with FPGA cards that include DDR4[5] and HBM2[6] memory have been used to validate this statement.

By temporarily storing the 1D FFT results for k consecutive rows in internal memory (e.g., Block RAM), with k being equal to the number of samples fitting within the access width of the DDR4 DIMM or HBM2 memory channel (e.g., k=4 64-bit samples would fit in a 256-bit wide access vector to the HBM2 memory), the transpose can be performed on the fly when writing the k row 1D FFT results back to the DDR4 or HBM2 memory (*Fig. 10* shows this procedure). The same
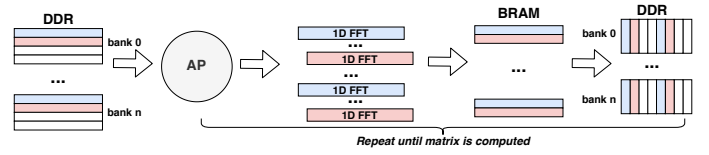

Fig. 10: `Access Processor`*'s data layout and processing for 1D FFTs over the rows showing the memory bank parallelism. This step is performed twice in order to obtain a 2D FFT.*

TABLE IV: *Estimated execution time of Access Processor for a single 2D FFT.*

Size	1 DDR4 DIMM 15 GB/s	2 DDR4 DIMM 30 GB/s	1 HBM2 channel 10 GB/s	32 HBM2 channels 320 GB/s
4 k	0.033 s	0.017 s	0.05 s	0.0016 s
8 k	0.13 s	0.067 s	0.20 s	0.0063 s
16 k	0.53 s	0.27 s	0.80 s	0.025 s
32 k	2.1 s	1.1 s	3.2 s	0.10 s

operation as described above is then repeated for the columns to obtain the overall 2D FFT results over the matrix. The effective memory access bandwidth is measured to be equal to $15\,\mathrm{GB/s}$ for a single DDR4 DIMM and $10\,\mathrm{GB/s}$ for a single HBM gen2 channel (which are conservative values also including the estimated impact of refresh operations, FPGA speed limitations, etc.), then the following execution times can be derived for the computation of the following four different 2D FFTs using DDR4 and HBM2 memory:

As shown in *Fig. 1* 2D FFT is the main bottleneck in IDG when enlarging the image size. We evaluate the benefits of applying NMC to FFT and comparing it to a Von-Neumann architecture. More precisely, we offload the 2D FFTs and their inverses to the `AP` design and to the NVIDIA V100.

We show how a NMC approach can be faster than a common CPU (see *Fig. 11*) outperforming it up to 120x. The proposed design can reach similar performance compared to a high-end GPU using less memory and having a maximum bandwidth lower than half. Furthermore, the FPGA has a lower thermal design power (TDP) compared to CPU and GPU, as reported in the data-sheets[1,2], which is 25W for the DDR4 board and 150W for the HBM2 board. Indeed, the IBM Power9 in use consumes around $480\,\mathrm{W}$ when performing the FFT and the NVIDIA V100 around $170\,\mathrm{W}$. We extract the power consumption with AMESTER[7] tool on the Power9 system including the NVIDIA V100. Thus, making FPGAs as good candidates for accelerating radio-astronomy applications.

V. RELATED WORK

In this section we provide the related work on workload characterization (*V-A*) and on the acceleration of 2D FFT kernels (*V-B*).

A. Application Characterization

A large amount of research has been focused on how to characterize workloads to detect bottlenecks. Yasin et al. [12]
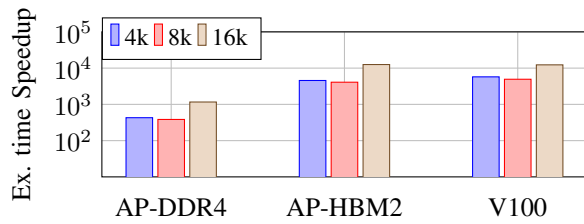

Fig. 11: *NMC-based platform and NVIDIA V100 execution time speedup compared to IBM Power9.*

presented a similar approach to the one used in this work, but on Intel systems being the foundations of the well-known Intel VTune [13]. It consists of a top-down approach to identify architectural bottleneck using selected PMUs. Awan et al. use that approach to spot architectural bottlenecks in big data applications [20], [21]. Differently, other approaches have been studied to characterize the application to be independent of the hardware. Corda et al. [22], [23] analyzed application at LLVM-IR level to extract intrinsic application features focusing on NMC. However, PMUs are faster to be used and more accurate. As side-effects PMUs are strictly dependent on the HW employed.

B. Large 2D FFT acceleration

Fast Fourier Transform is one of the most widely studied algorithms in the past. Especially, large 2D FFT that is expensive on CPU, because of the enormous amount of data that must be moved from main memory through the cache hierarchy and vice-versa, have been improved. Dang et al. [24] proposed an FFT implementation on GPU clusters applied to large electromagnetic problems. Yu et al. [25] and Akin et al. [26] developed two tiling algorithms to improve performance on the 2D FFTs on different platforms. Differently from the previous work, we employed a new computational paradigm called near-memory computing and we focused on larger 2D FFT sizes applied to radio-astronomy imaging.

VI. Conclusion

We analyzed the state-of-the-art gridding and degridding imaging algorithm for radio-astronomy, as used in SKA, the largest radio telescope on Earth. We employed the CPI breakdown analysis and the roofline model on IBM Power9 identifying the memory bottlenecks. Then, we showed how these bottlenecks can be alleviated by applying an NMC approach to FPGA and comparing it to CPU and GPU. Thus showing how an NMC approach can highly outperform a CPU and can achieve similar performance compared to a high-end GPU, which has higher memory bandwidth and memory size.

Acknowledgments

References

[1] R. V. van Nieuwpoort *et al.*, "Correlating radio astronomy signals with many-core hardware," *International Journal of Parallel Programming*, vol. 39, no. 1, p. 88114, Jun 2010.

[2] B. Veenboer *et al.*, "Image-domain gridding on graphics processors," in *2017 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2017, Orlando, FL, USA, May 29 - June 2, 2017*, 2017, pp. 545–554.

[3] ——, "Radio-astronomical imaging: Fpgas vs gpus," in *Euro-Par 2019: Parallel Processing - 25th International Conference on Parallel and Distributed Computing, Göttingen, Germany, August 26-30, 2019, Proceedings*, 2019, pp. 509–521.

[4] A. J. Awan, "Performance characterization and optimization of in-memory data analytics on a scale-up server," Ph.D. dissertation, KTH Royal Institute of Technology and Universitat Politècnica de Catalunya, 2017.

[5] W. A. Wulf *et al.*, "Hitting the memory wall: Implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, pp. 20–24, Mar. 1995.

[6] H. Esmaeilzadeh *et al.*, "Dark silicon and the end of multicore scaling," *SIGARCH Comput. Archit. News*, vol. 39, no. 3, pp. 365–376, Jun. 2011.

[7] G. Singh *et al.*, "A review of near-memory computing architectures: Opportunities and challenges," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, Aug 2018, pp. 608–617.

[8] ——, "Near-memory computing: Past, present, and future," *Microprocessors and Microsystems*, vol. 71, p. 102868, 2019.

[9] ——, "Napel: Near-memory computing application performance prediction via ensemble learning," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[10] A. R. Offringa *et al.*, "Wsclean: an implementation of a fast, generic wide-field imager for radio astronomy," *Monthly Notices of the Royal Astronomical Society*, vol. 444, no. 1, p. 606619, Aug 2014.

[11] R. F. Araujo *et al.*, "A cpi breakdown model plug-in for optimizing application performance," in *2013 3rd International Workshop on Developing Tools as Plug-Ins (TOPI)*, May 2013, pp. 31–36.

[12] A. Yasin, "A top-down method for performance analysis and counters architecture," in *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, March 2014, pp. 35–44.

[13] INTEL. Intel vtune amplifier. [Online]. Available: https://software.intel.com/en-us/intel-vtune-amplifier-xe

[14] J. v. Lunteren *et al.*, "Coherently attached programmable near-memory acceleration platform and its application to stencil processing," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019, pp. 668–673.

[15] ——, "Designing a programmable wire-speed regular-expression matching accelerator," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec 2012, pp. 461–472.

[16] Linux. Perf events tutorial. [Online]. Available: http://perf.wiki.kernel.org/

[17] S. Williams *et al.*, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 6576, Apr. 2009.

[18] H. Giefers and et al., "Accelerating arithmetic kernels with coherent attached fpga coprocessors," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE 15. San Jose, CA, USA: EDA Consortium, 2015, p. 10721077.

[19] B. Sukhwani *et al.*, "Contutto a novel fpga-based prototyping platform enabling innovation in the memory subsystem of a server class processor," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2017, pp. 15–26.

[20] A. J. Awan *et al.*, "Identifying the potential of near data processing for apache spark," in *Proceedings of the International Symposium on Memory Systems*. ACM, 2017, pp. 60–67.

[21] ——, "Micro-architectural characterization of apache spark on batch and stream processing workloads," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud)*. IEEE, 2016, pp. 59–66.

[22] S. Corda *et al.*, "Memory and parallelism analysis using a platform-independent approach," in *ACM 22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES '19)*. Sankt Goar, Germany: ACM, May 2019.

[23] ——, "Platform independent software analysis for near memory computing," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, Aug 2019, pp. 606–609.

[24] V. Dang and et al., "GPU cluster implementation of FMM-FFT for large-scale electromagnetic problems," *IEEE Antennas and Wireless Propagation Letters*, vol. 13, pp. 1259–1262, 2014.

[25] C. L. Yu *et al.*, "FPGA architecture for 2d discrete Fourier transform based on 2d decomposition for large-sized data," *Journal of Signal Processing Systems*, vol. 64, no. 1, pp. 109–122, 2011.

[26] B. Akin *et al.*, "Memory bandwidth efficient two-dimensional fast Fourier transform algorithm and implementation for large problem sizes," *Proceedings of the 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines, FCCM 2012*, pp. 188–191, 2012.