Theme 8 - Characterisation of HW Platforms and Accelerators

Introduction

The purpose of Theme 8 was to concentrate on Hardware Platforms.

The initial topic was Heterogeneous Computing Platforms for SKA Data Processing. This topic was very broad and after discussions within the team it was decided that the expertise that existed at within the group it would be best to refine this to look at acceleration of specific parts of the radio astronomy data processing pipelining.

The area of data storage was not covered in the end as the topic proposal for that area, although relying on characteristic of HW platforms, was at the stage of being an initial software implementation.

Acronyms

Acronym	Definition
FFT	Fast Fourier Transform
FPGA	Fully Programmable Gate Array
MIC	Many Integrated Cores
MWA	Murchison Widefield Array
SDP	Science Data Processor

Defining the Collaboration Task

Each person within the team took it in turn to describe what their workshop topic was. This gave the members of the team an idea of each person's likely expertise and area of interest. As many of the team members had FPGA, MIC or Embedded System expertise along with published papers already on FPGA acceleration of Fast Fourier Transforms the patch of obvious research as a team came out to be acceleration of specific pipeline component kernels.

Discussions with other experts in the room, thank you particularly to Andreas Wicenec, the areas that seemed feasible to include were considered to be Gridding, FFT and Convolution.

A point raised quickly by one of the observers were concerns related to, in particular FPGAs, about the use of the accelerator as requirements and workloads change. This feedback was taken into account whilst defining the requirements for the platform.

Deciding on the required area of research then naturally lead to discussions about how to show such systems could be used in an astronomical pipeline and then scaled to SKA levels. The basis for a Test Suite (benchmarking HW platform) was discussed with particular focus on what could be measured to show the system was capable of delivering. Essentially a comparison of performance and power usage for each of the options, including a “standard” x86 CPU only,

Hardware Options

The “standard” platform for running the pipeline software is based on the x86 CPU. Various filesystem (local and global) plus high speed networking are integrated into these platforms with performance characteristics typically matching the number of nodes and performance of the nodes.

The following subsections discuss the HW requirements of the HW acceleration options.

FPGA

The requirements of an FPGA based acceleration option

1. FPGA based accelerators are expected to support both PCI-E and Ethernet as I/O channels. This allows the FPGA to communicate with CPU on the same computing node via PCIe bus while FPGA can also transfer input and output data to computing resources on other computing nodes
2. High level programming languages are preferred as design tool of FPGA based accelerators, such as OpenCL C or C++.
3. Dynamic configuration of FPGA is needed to support switching of multiple function configuration.
4. High bandwidth DDR3/4 is required to work as on board memory

MIC

The selection of a MIC is much easier and there is really on the Knights Landing product currently.

The current generation of MIC (Knights Landing)

1. Bootable host processor & Coprocessor
2. up to 72 cores, 288 threads
3. 3 TFLOP/s DP, 6+ TFLOP/s SP
4. Up to 16GB on-package MCDRAM> 400GB/s sustained BW
5. 2x 512b VPU pre core, supported AVX-512
6. Binary compatible with Intel Xeon
7. Integrated Intel Omni-path

GPU

Although a valid accelerator option practicalities of limited resources mean that GPU categorisation will not be included.

Algorithms to Test

Computationally expensive but well understood (reference implementations available) areas were chosen as the most likely test candidates.

Gridding

FFT

Convolution

Supporting Work

FFT on FPGAs

Yongxin Zhu's team at Shanghai Jiao Tong University has made prototyping implementation of FFT on FPGAs. They demonstrated variable precision in their FFT implementations by both implementing a new floating point numeric representation (Universal Number) and fixed point approximation of floating point numeric, whose speedup is over 20x compared with CPU as reported in their publication in FPT Conference (Field Programmable Technologies, one of top 4 FPGA conferences) 2016:

Mengjun Li, Yongxin Zhu*, Xu Wang, Tian Huang. Evaluation of Variable Precision Computing with Variable Precision FFT Implementation on FPGA, The 2016 International Conference on Field-Programmable Technology (FPT '16), China, December 10-12,2016.

Gridding on FPGAs

Yongxin Zhu's team at Shanghai Jiao Tong University also prototyped implementation of Gridding on FPGAs. They showed that FPGA based implementation of gridding algorithm is feasible even on a student FPGA board from Xilinx. Their initial results including a speedup over 8x compared with CPU are shown in their paper to appear in FCCM Conference (Field Customized Computing Machine, another top 4 FPGA conferences) 2017:

Qian Wu, Yongxin Zhu*, et al. Exploring High Efficiency Hardware Accelerator for the Key Algorithm of Square Kilometer Array Telescope Data Processing[C]. Field-Programmable Custom Computing Machines (FCCM), 2017 IEEE 25th Annual International Symposium on. IEEE, 2017.

Further implementation with bigger data sets are being prototyped on self-made FPGA boards which contain huge computing capacities as reported in Journal of Systems Architecture:

Wang X, Zhu Y*, Ha Y, et al. An energy-efficient system on a programmable chip platform for cloud applications[J]. Journal of Systems Architecture, 2016.

Gridding on MICs

Shaohua Wu's team from Inspur(Beijing) Electronic Information Industry Co., Ltd. has implemented the Gridding algorithm on MICs which include KNC(Knights Corner) and KNL(Knights Landing). The results on KNC gained a speedup over 5.2x compared with the CPU benchmark version. Two versions of the optimized codes have been submitted on Confluence. On KNL, we have also gotten over 12x speedup compared with the initial code.

We have built a KNL cluster which has 64 nodes and detailed information about it is given as follows.

Nodes	64
KNL	64*Intel(R) Genuine Intel(R) CPU 0000 @ 1.3Ghz, 16GB MCDRAM, DDR 2133
Storage	IEEL
Network	OPA
OS	Red Hat Enterprise Linux Server release 7.1(Maipo)
Compiler	icc,icpc,ifort
MPI	Intel(R) MPI Library for Linux* OS, Version 5.0 Update 1 Build 20140709
Tool	Intel Parallel Studio 2017

We have also protomed KEEP which is short for Knights Landing Evaluation and Escalation Program. KEEP was official joint launching by Inspur and Intel at ISC2016, which is open to the HPC community and users from academic, industry and enterprises. By applying online through http://inspurhpc.com/KEEP , you can Participate KEEP and use the KNL cluster for free. Please contact KEEP@inspur.com or contact donghao@inspur.com if you have any questions.

Performance Benchmarking

The SDP has already done some work in this area (Product Tree C3.2.5 Workload Characterisation Framework). This will be used as the starting point for determining the what and how of the workload performance.

In addition input from various groups within the other workshops groups was gathered to reduce the risk of not taking into account metrics that some may consider crucial to justifying the final solution.

Execution Framework

It is well understood that general purpose processor (CPU) can takes an advantage of its strong compatibility to make logical control of an execution pipeline of SKA-SDP. However, CPU usually has lower performance compared with heterogeneous accelerators such as GPU/FPGA/MIC, among which FPGA has much higher power efficiency.

A heterogeneous computing framework is expected to consist of general purpose processors on a hosting server with heterogeneous accelerators including FPGA/MIC/GPU to combine the advantages of both processor and accelerators.

X-86-Based Server + FPGA-Based Accelerator Boards

A heterogeneous computing architecture consisting of X-86-Based sever + Xilinx FPGA is a candidate of “execution framework” prototyping. The framework including FPGA is shown as follows.

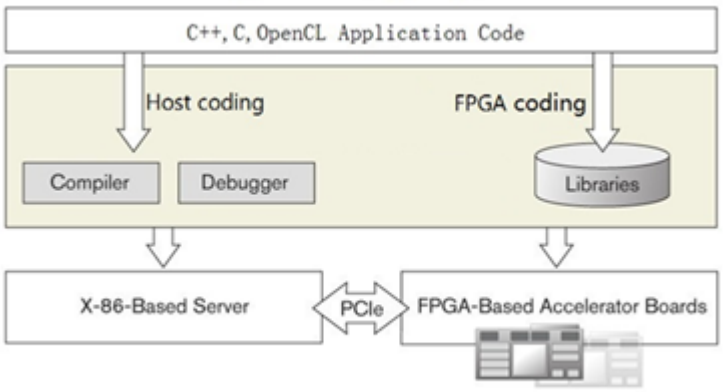

Fig. 1 An FPGA based candidate of execution framework.

Xilinx provides SDAccel development environment supporting C++, C and OpenCL (Open Computing Language, which is based on C99, so is very similar with C++) development on FPGA. The application code consists of two parts, host code (which can be coded by OpenCL, C, C++) and kernel code (which can be coded by OpenCL, C, C++, Verilog and VHDL), as shown below.

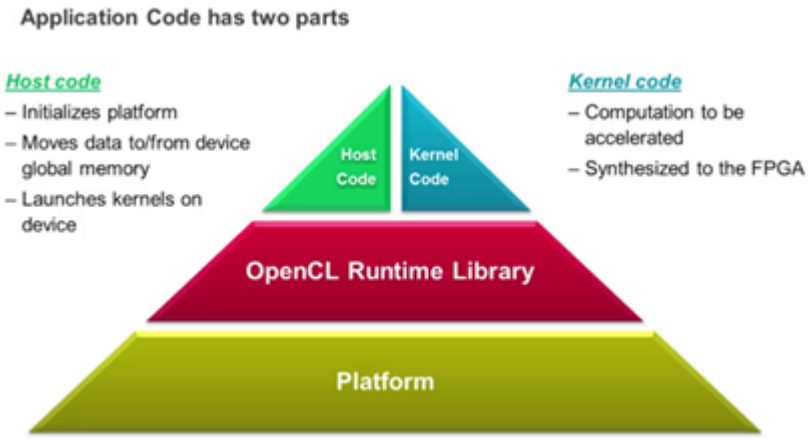

Fig. 2 Software-hardware co-design tool chain

Host code has three main functions: initializes platform, transfers data by PCIe port and launches kernels on FPGA. Kernel code includes the part to be accelerated and is synthesized to FPGA. With SDAccel development environment, a heterogeneous computing is not such difficult like before. In SKA data process, some logical control task should be allocated to host processor, such as data preprocess. This part can be easily coded by C++ and run on host processor. Some data-intensive task or high parallel computation, such as Gridding, FFT should be implemented on FPGA by kernel code.

To reuse existing prototyping results, we take Wei Wang's COTS (commercial off the shelf) prototyping work [1] on big data platform SPARK as an example to show the overall data flow in SKA and the execution time at every stage, as shown as follows.

Stage ID	**Function**	**Time**
0	local_sky_model	**2s**
1	extract_lsm	**2s**
2	telescope_management	**2s**
3	telescope_data	**0.2s**
4	reprojection_predict_ifft	**5.1m**
5	degridding_kernel_update_degrid	**29m** original
6	extract_lsm	**2s**

7	phase_rotation_predict_dft_sum_visibilities	**53s**
8	visibility_buffer	**45s**
9	Timeslots	**1.7m**
10	phase_rotation_predict_dft_sum_visibilities	**36s**
11	visibility_buffer	**29s**
12	Solve	**15s**
13	correct_subtract_visibility_flag	**2.7m**
14	telescope_data	**0.2s**
15	gridding_kernel_update_phase_rotation_grid_fft_reprojection	**3.6m**
16	sum_facets	**1.3m**
17	identify_component	**1.3m**
18	source_find	**0.05s**
19	Println	**0.05s**

Table 1. Computation overheads of each pipeline stage in COTS execution prototyping

The table above shows the most expensive stages: stage 4 reprojection_predict_ifft (5.1 minutes), stage 5 degridding_kernel_update_degrid (29 minutes) and stage 15 gridding_kernel_update_phase_rotation_grid_fft_reprojection (3.6 minutes). In other words, (i)FFT and (de)Gridding have longest execution time, and their computation formulas indicate they can be implemented by pipelined and parallel architecture.

$$\mathrm{X(k)} = \sum_{n=0}^{n=N-1} x(n) e^{-j\left(2pi * \frac{nk}{N}\right)}, \mathrm{n} = 0, 1, 2, \ldots, \mathrm{N\text{-}1}$$

$$\mathrm{grid(u,v)} = \sum_{\mathrm{x=-support}}^{\mathrm{support}} \sum_{y=-support}^{support} Samples(u-x, v-y) \times C(-x,-y)$$

So, the three stages should run on FPGA acceleration board in the proposed heterogeneous architecture. The overall data flow is distributed as follows. Stage 4, 5, 15 runs on FPGA

accelerated board and others run on server. Data transmission between server and FPGA is through PCIe temporarily.

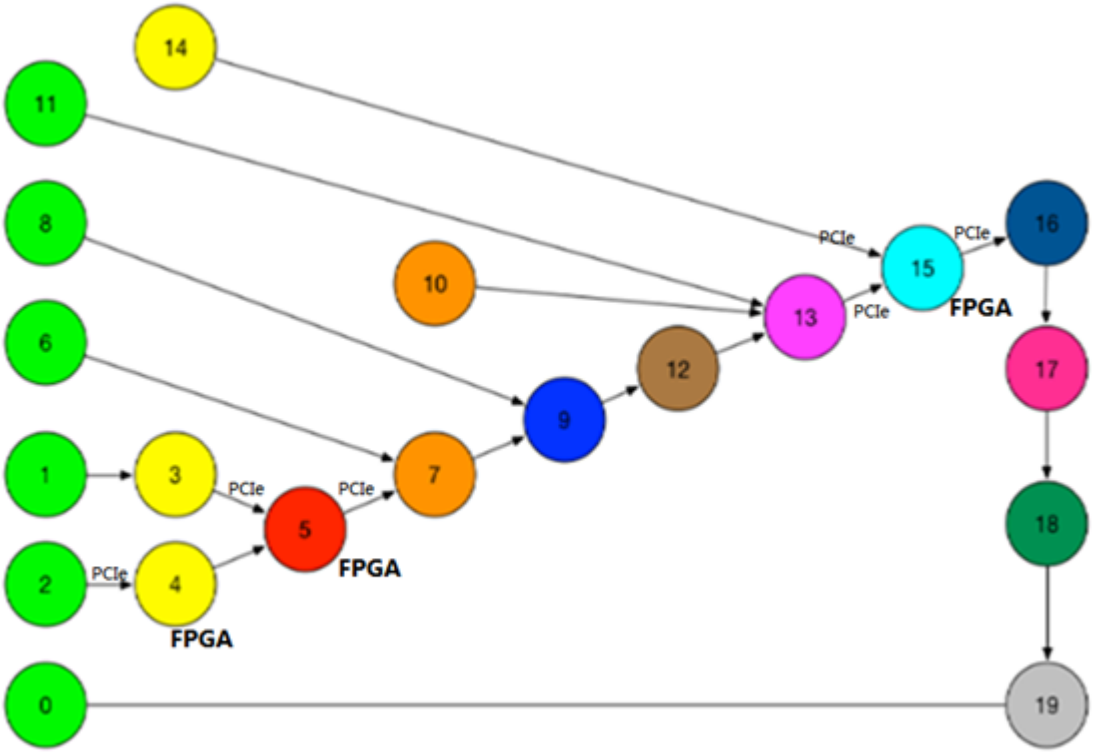

Fig. 3 Heterogeneous execution of SKA-SDP pipeline

Discussion about performance and scalability issues.

1. The data can be transmitted by PCIe (maximum transmission speed is 64 Gbps). But, if PCIe transmission bandwidth is inadequate, Ethernet port or optical port (maximum transmission speed is up to 300 Gbps) on FPGA can be another input source.

2. If the source on a FPGA accelerated board is not enough for data processing, we can divide the whole task into small tasks running on each heterogeneous computing unit, which can be named as “multiply hosts multiply boards”, or add more boards connected with host server by PCIe or Ethernet port, which can be named as “one host multiply boards”.

Reference

https://confluence.ska-sdp.org/display/WBS/COTS+Report

Test Data Set

The choice of data is limited to that in the public domain.

As the team investigating RFI have already provided data from the Australian Compact Arrary, it is believed this would be a good starting position.

Longer term access to precursor data (MWA perhaps?) would help validate any solution as being worthy of further investment.

Pipeline Software Integration

Proving that a small part of the pipeline SW can be optimised showing performance and/or power consumption is important.

Capturing and Analysing the Results

Benchmarking and Monitoring

The work so far on SDP provides mechanisms for monitoring the CPU for utilisation, power etc. It was identified though that additional HW support would be required to provide a consistent mechanism to monitor power.

The following were identified as areas to be monitored/tested

1. Data gathered from hardware to provide clear performance metrics
    a. Power consumption
    b. System utilisation
    c. Time passed (start - end of operation)
2. Algorithm performance
    a. Time based
    b. Resources used
    c. Good science result
3. Result output (“standard” vs accelerator) must be consistent
    a. What variation of result is acceptable?

Workplan

The following outlines how the team will go about developing, benchmarking and validating the goal algorithms. This needs refining into smaller chunks that can be achieved in 6 month chunks based on available funding, resources and expertise.

1. Communicate with pipeline experts to ensure validity of test cases
2. Further refine the required HW to support the algorithms to be tested as well as the benchmarking and testing methodologies
3. Seek funding to procure several systems to use as the platform for development and testing.
4. Develop the required algorithms to operate on the accelerator
5. Investigate/develop methods to integrate HW accelerated algorithms into pipeline so test against data sets can be validated.

Results

The following results will be recorded

1. Define a heuristic to represent performance and power for tested metrics
2. Develop a matrix of algorithms vs. hardware platforms
    a. Power
    b. Performance
    c. …..
3. Results of integration of accelerated algorithms into pipeline