IJCSE International Journal of Computer Science and Engineering Open Access
Technical Paper Volume-2, Issue-3 E-ISSN: 2347-2693

Multicore Heterogeneous Computing with OpenACC

Niraj R Chauhan[1*] and Mayur S. Burange[2]

[1*,2]*Dept. of Comp. Sc & Engg., P.R.Pote Engg & management College, Amravati - India*
nirajchauhan.nri24@gmail.com; mayurmsb123@gmail.com
www.ijcaonline.org

Received: 05 Mar 2014 Revised: 14 Mar 2014 Accepted: 24 Mar 2014 Published: 31 Mar 2014

Abstract- OpenACC it is a standard programming language for programming heterogeneous computers built from CPUs, GPUs and DSP. It includes a framework of OpenAcc to define the platform in terms of a host (e.g. a CPU) and one or more graphical compute devices (e.g. a GPU) plus a C-based programming language for writing programs & for executing program for the computer devices. Using OpenAcc a programmer can write task-based programming and data-parallel programming that are use all the resources of the heterogeneous computer system. This will be a "future introduction to programming language" where we cover the ideas behind OpenAcc & other platforms. Thereby providing a pedagogically useful example that experienced heterogeneous computing programmers will need to quickly become productive & efficient OpenAcc programmer's model. We can also show how these ideas are translated into source code & how they are executed in the given system. We will do this through a series of progressively more challenging examples for heterogeneous computing system.

Keywords- OpenAcc, Heterogeneous computing, HAS, OpenCL, CUDA

HETEROGENEOUS COMPUTING WITH OPENACC

Heterogeneous computing refers to systems of CPU, GPU & others processor, which are use more than one kind of processor. These are multi-core heterogeneous computer systems that gain performance that are not just by adding cores, but also by including specialized processing & manipulating capabilities to handle many particular tasks at random time. This document describes the OpenAcc compiler directives, library routines and environment variables that collectively define the OpenAcc Application Programming Interface (OpenACC API) for off-line programs written in C, C++ and Fortran programs from a host computer system CPU to an attached accelerator device with accelerator processor. This method provides a model for accelerator programming that is portable & present across operating systems and various types of host computer system CPUs and accelerators processor. The directives extend the ISO/ANSI standard C, C++ and Fortran base languages & many more, in a way that allows a programmer to integrate applications incrementally to accelerator targets & aim of system using standards-based C, C++ or Fortran. Heterogeneous Computer System Architecture utilize multiple multicore processor types, usually on the same silicon die, to give you the best of both worlds: GPU processing, apart from its well-known 3D graphics rendering capabilities & execution, can also perform mathematically intensive computations on very large data sets, while CPUs can run the operating system and perform traditional serial tasks.

On the other hand, the multiple multi-core era also completed some interesting developments in GPUs, which were growing in sophistication and complexity, spared on by advances in semiconductor Software technology. GPUs & DSP have vector processing capabilities which them enable to perform parallel operations on very large scale sets of data – and to do it at less power consumption relative to the serial processing system of similar data sets on CPUs. By using this,GPUs to drive capabilities such as incredibly realistic, multiple display stereoscopic gaming & various forecasting system such as

Weather, nature, aerospace. And while their value was initially derived from the ability to improve 3D graphics performance by offloading graphics from the CPU, so that they became increasingly attractive for more GPUs, such as locating data parallel programming tasks.

The directives, packages and programming model defined in framework of openAcc, allow programmers to create applications which are capable of using accelerators processor, without the need to explicitly manage data or program transfers between two host and accelerator computer system. Rather, these details are implicit in the programming model of OpenAcc and are managed by the OpenAcc API-enabled compilers and runtime environments System. The programming model of openAcc allows the programmer to augment information available to the compilers, including specification of data local to an accelerator, guidance on mapping of loops onto an accelerator, loops of various methodical and similar for performance-related details.

SCOPE OF OPENACC FOR HETEROGENEOUS COMPUTING

This OpenAcc API document covers only user-directed accelerator programming, where the user specifies the regions of a host program to be targeted for offloading to an accelerator device. The main point of the program will be executed on the host CPU system. This document does not describe features or limitations of the host programming environment as a written by programmer; It

Corresponding Author: Niraj R Chauhan[1]

is limited to specification of various loops and regions of code to be offloaded to an accelerator.

Various challenge with heterogeneous computing with OpenAcc which are discussed below.

- Levels of parallelism:
  There are many levels of parallelism for heterogeneous computing system
  - ✓ multiple device are inserted with separate (physical) address spaces & locations
  - ✓ multiple threads are created inside a device
  - ✓ vector parallelism is done by programming model inside the thread mechanism
- Types of parallel programming models:
  - ✓ Message Passing interface which is used for distributed or clustered memory systems
  - ✓ Task parallel interface which is used for shared memory systems, thread based system
  - ✓ Data parallel interface used for streaming processing system

- Exploiting heterogeneous environment:
  - ✓ It is studied by GPGPU APIs and various programming languages like OpenCL, CUDA
  - ✓ It included various Libraries such as CUBLAS, CUSPARSE, MAGMA etc
  - ✓ It included various Compiler directives (PGI Accelerator Directives, OpenACC)

This document does not describe automatic detection and offloading of regions of code to an accelerator by a compiler or other tool. This document does not describe splitting loops or code regions to multiple accelerators attached to a single host. While future compilers may allow for automatic offloading, or offloading to multiple accelerators of the same type, or to multiple accelerators of different types, these possibilities are not addressed in this document

OPENACC EXECUTION MODEL

We know that OpenAcc programming model targets on a host-directed programming execution model where the sequential programming code runs on a conventional special purpose processor and computationally integrated data & task parallel pieces of code (kernels) run on an accelerator processor such as a CPU,GPU & other processor . This mapping between the host computer system and data & task parallel devices reflects a hardware & software expression specification of Amdahl's Law.

Amdahl's Law is named after the computer architect scientist Gene Amdahl. It is an approximation of that models the ideally speedup sequential processor which are happen when sequential (single-threaded) programs & parallel programming model are modified to run on parallel hardware. In the ideal case, those sections of code that can be parallelized can have their runtime reduced by a factor of N, where N is the number of parallel processing elements. Large application speedups can occur when the parallel sections of code dominate the runtime of the sequential code. Theoretically, the time taken to complete the serial sections of code (those sections that cannot be parallelized) will eventually decreases the runtime scheduling of processor, when the number of task parallel processing elements are allowed for performing various task at single time. Where N, is large.

There are various benefit of the OpenAcc host-directed heterogeneous programming model that it can leverage the massive parallelism of heterogeneous system of one or more accelerator devices where the number of processing elements are large, while preserving these ability of the latest generation of heterogeneous sequential processors to accelerate the serial sections of code. In this way, the performance characteristics of both art processors and accelerators can be expanded in near future.

OpenAcc programmers must have to pay attention to data transfers and usage of OpenAcc heterogeneous devices. High-performance application interface generally conform to the following three rules of co-processer programming models:

- Transfer the data across the various PCIes bus onto the heterogeneous device and keep it there for future purpose use.
- Give the device enough work to do so that performance of system must be increased.
- Focus on data reuse within the co-processors to avoid memory bandwidth bottlenecks problems in heterogeneous.

Kernel regions operating system bundle one or several nested loops of threads into a kernel memory space. The OpenAcc compiler system essentially translates the loops of processing threads into a kernel that can run in parallel environments on the accelerator processing system. Today's present compilers are able to offload loops that are nested three levels in deep. When the nesting level exceeds the compiler capability, the outer loop(s) will run sequentially on the any host it may be heterogeneous processing, while the inner loops will run on the accelerator processing unit. When possible, it is most importantly ensure that the inner loops of processing perform the largest amounts of work on host devices to make best use of the accelerator processing unit. Following is an example of a nested set of matrix multiplication loops that was allotted as a kernel region for the OpenAcc compiler:

Compute matrix multiplication.

```
#pragma acc kernels copyin(a,b) copy(c)
 for (i = 0; i < SIZE; ++i) {
  for (j = 0; j < SIZE; ++j) {
   for (k = 0; k < SIZE; ++k) {
  c[i][j] += a[i][k] * b[k][j];
   }
  }
 }
```

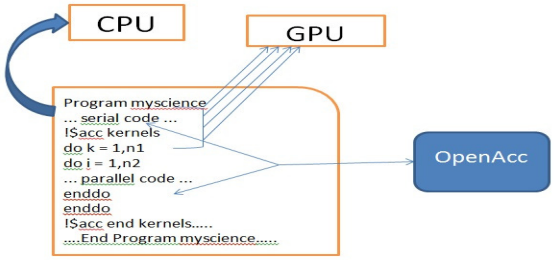

Figure 1

OpenAcc programming model targets a general purpose device architecture that consider that a device will have to contain multiple processing elements which are run in parallel programming model system. Each processing elements mentioned that they has the ability to efficiently & correctly perform vector-like operations & manipulation on various operation. For AMD GPUs, it is mandatorily to think that each processing elements as a streaming multicore multiprocessor, which are represented by OpenAcc as thread block, that is a *worker* which is effectively a warp, and that an OpenAcc *vector* is a form of OpenCl thread & CUDA thread.

MEMORY MODEL OF OPENACC

OpenAcc is mainly used for targeting various heterogeneous computer systems with a host CPU computer system and a attached accelerator computer device so that heterogeneous computer system nature is maintain between them , such as an GPU,APU, the memory space of the host computer system and accelerator computer system is mandatorily completely separately & individually . All data allocation and data movement between host memory and accelerator device memory must be performed by the host through runtime library calls. Besides the host computer device and accelerator computer device ,the device memory of OpenAcc also has the notion of private, protected and shared memory, where the private memory & protected memory is generally use for hardware-managed caches and the shared memory is usually used for software-managed cache like the shared memory on OpenCl, CUDA. All of these memory concepts are implicitly and manageable by the compiler of openAcc, based on compiler directives declared by the system programmer.

The directives of openAcc can describe the allocation and data movement by telling the compiler that memory should be allocated in device memory, data should be copied from the host CPU computer system to the heterogeneous computer systems and vice versa. Only copied data from the host computer systems to the heterogeneous computer systems device, only copied to the CPU computer systems host from the heterogeneous computer systems device, or that the data already exists in the device memory of both computer system.

In a typical data parallel region, it is copied to the device from the host CPU computer systems at the start of the execution, and copied back to the host CPU computer systems when the execution of program ends. On the other hand, when there are multiple parallel regions of data & task that all work on the same data field which are present in sequence of the copying of data back and forth between each region of execution of programming data which is might be slow down performance of CPU computer systems. It is therefore possible to use the OpenAcc runtime API to allocate memory at the start of the program and pass the device memory pointers to the parallel regions of CPU computer systems, allowing the data to stay on the CPU computer systems device through the parallel programs execution in the heterogeneous computer architecture system.

In addition the compiler of OpenAcc also creates barrier constructs to prevent simultaneous access of memory space to the same memory location that might be result in memory coherence issues, which is repeatedly represented by this computer system.

The concept of separate host CPU computer systems memories and accelerator computer systems memories is very apparent & general in low-level accelerator programming languages such as CUDA or OpenCL, in which data transformation between the memories can be dominate & reutilizing user programming code . In the OpenAcc programming model, data movement & transformation between the memories of host CPU computer systems memories and accelerator computer systems memories can be implicitly and managed by the compiler, based on directives from the programmer. However, the programmer must be aware of the potentially separate memories for many reasons, including but not limited to:

- Memory bandwidth between CPU computer systems memories and accelerator computer systems memories determines & logically use the level of compute intensity required to effectively integrated & accelerate a given region of programming code for OpenAcc framework
- The limited device memory size may prohibit offloading of regions of programming code that operate on very large scale amounts of data which is present in between CPU computer systems memory and accelerator computer systems memory.
- Host CPU computer systems addresses stored to pointers on the host CPU computer systems& it may only be valid on the host computer systems. They may addresses stored to pointers on the device may only be valid on the device of host. Dereferencing host pointers on the device or dereferencing device pointers on the host is likely to be invalid on such targets memory system.

Device data has an explicit lifetime, from when it is allocated or created until it is deleted. If the device shares physical and virtual memory with the local thread, the device data environment will be shared with the local thread. In that case, the implementation need not create new copies of the data for the device and no data movement need be done. If the device has a physically or

virtually separate memory from the local thread, the implementation will allocate new data in the device memory and copy data from the local memory to the device environment.

OpenAcc exposes the separate memories through the use of a device data environment. Some accelerators (such as current GPUs) implement a weak memory model. In particular, they do not support memory coherence between operations executed by different threads; even on the same execution unit, memory coherence is only guaranteed when the memory operations are separated by an explicit memory fence. Otherwise, if one thread updates a memory location and another reads the same location, or two threads store a value to the same location, the hardware may not guarantee the same result for each execution. While a compiler can detect some potential errors of this nature, it is nonetheless possible to write an accelerator parallel or kernels region that produces inconsistent numerical results. Some current accelerators processing units some have hardware managed caches system & software-managed cache system, and most have hardware caches that can be used only in certain condition and are limited to read-only data in CPU computer system. In low-level programming models such as CUDA or OpenCL languages, it is up to the programmer to manage these caches. In the OpenAcc model, these caches are managed by the compiler with hints from the programmer in the form of directives.

The possibilities created by heterogeneous computing are truly fantastic – from flawless HD videoconferencing to heretofore unimaginable display clarity to real-time language translation and interpretation – all in lower and lower power envelopes for smaller and smaller form factors with longer and longer battery life.

WORKING OF HETEROGENEOUS SYSTEMS

It is currently perform leverage of the power efficiency and data & task parallelism of today's heterogeneous to user in era of innovation in heterogeneous computing. So that they can take mainstream visual virtual desktop applications and they extended them to new levels, so that they enable new pixel intensive experiences and even introduce non visual & virtual capabilities where they are unimaginable & unusable – until up to date. So let's take moments to look at the many ways of using GPUs continue to propel advances in mainstream heterogeneous computing.

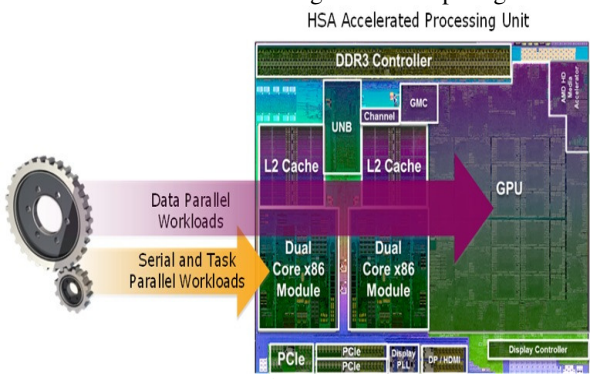

Figure 2

Way to accelerating application of heterogeneous computing with OpenAcc

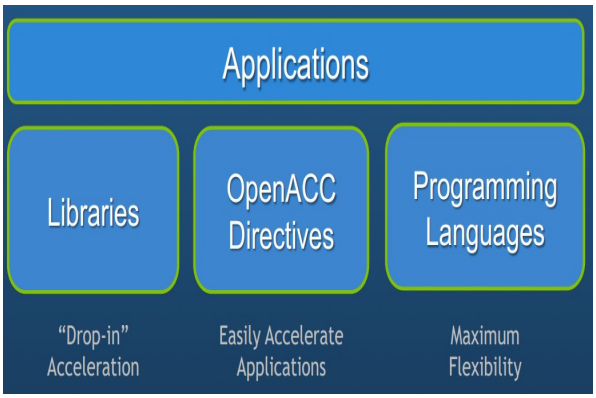

Figure 3

Today's GPUs can enable ultra-high frame rates for realistic, immersive 3D gaming on mainstream (even entry level) heterogenous computers. They can also make it easy and cost-effective to add various 3D stereoscopic realism to 2D content, that is from HI definition Hollywood movies to home video system. Today's GPUs can make both collaborative definition and independent definition of work for more productive result in productive industries. With support for flawless HD video conference & video which are capabilities and new forms of virtual presence of various architecture of heterogeneous computing system GPUs make virtual meeting as close to meeting in person as it gets.so that they enable visual virtual video communications that were not even remotely possible for communication before, like bidirectional heterogeneous HD video chats & communication. They can also send desktop productivity soaring by making it easier and more seamless than before to move back to and forth between applications on multiple monitors of host system & another system – even when working with graphics processing computing intensive content like PowerPoint presentations & product demos or simulations, and various graphical videos.

The performance of computing using heterogeneous computing is shown as below

.

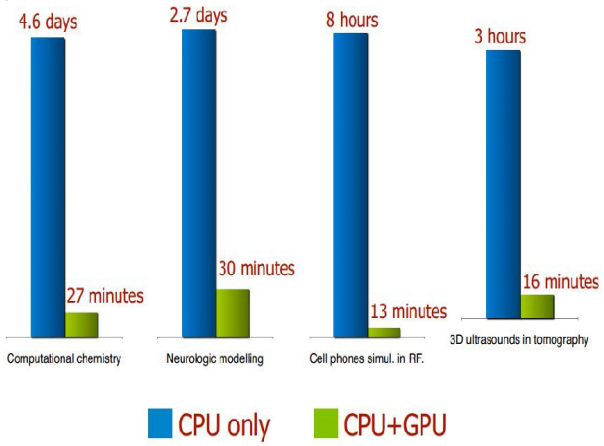

Figure 4

HAS(HETEROGENEOUS SYSTEM ARCHITECTURE) ENABLING NEW PIXEL-INTENSIVE EXPERIENCES

Those are just a few of ways that heterogeneous computing could change the way people interact with computers forever. The only thing more exciting than doing something better than you've ever done it is doing something entirely new. For example, what if you could log in to your computer just by looking at the screen? What if it could respond to just a gesture, not even need a touch, to know exactly what you want? And what if your HD video walkthrough of a new house could generate a 3D model of that space to see if your furniture could make that house your new home? Those are just a few of ways that heterogeneous computing could change the way people interact with computers forever. They're also just three examples of the types of software innovations that are possible now with GPU application programming interfaces (APIs).

If you're like most people, you probably find it hard to imagine something more mundane and time consuming than searching and categorizing your photos and videos based upon who is in them, where you were and what you were doing. But what if your computer could speed through by quickly and automatically finding and tagging photos and videos for you based on the faces, places or objects that are in the images? Or by helping you sort through photo libraries to eliminate duplicates saved under different names? Or by finding the IMDb or Wikipedia entry you're looking for based on a "look" at an actor's face? These are also pixel-intensive experiences made possible by today's GPUs. And they're enough to make current search and index capabilities seem almost unimaginably tedious and slow.

Today's GPUs can also add measurably to the enjoyment of visual content in a number of new ways. Imagine a computer that enhances your shaky handheld video footage by automatically stabilizing HD content. Or one that delivers crystal clear content even on room-sized screens. Or that supports holographic imagery for unprecedented realism. We foresee display resolutions so high, they deliver density and clarity that rivals what the human eye can even perceive in the analog world. This is how technology improves our lives by getting out of the way.

HAS CREATING NEW NON VISUAL EXPERIENCES

Chances are that when you think of new applications for GPUs, you're likely to think of visual experiences. Facial and gesture recognition, photo or video indexing, and some of the others we've just described are good examples. That stands to reason, given that the "G" in GPU does stand for "graphics". But as we have discussed, GPUs can do more than just push pixels to a display. They are very capable general purpose computing resources that can now stand side by side with the CPU and enable a broad range of compelling new experiences. For example, today's GPUs enable a level of contextually aware computing that takes new experiences far outside the realm of the visual. We may not be at a point where "Computer: Earl Grey, hot" is going to be a reality anytime soon. But some of the very real possibilities are nearly as amazing.

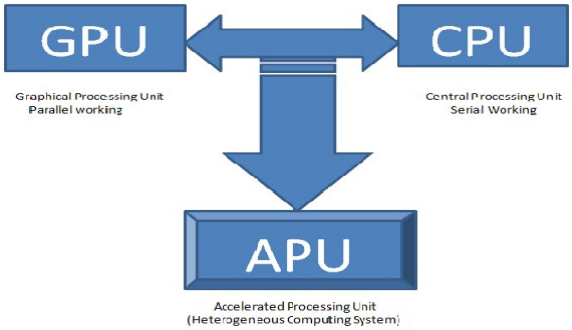

Figure 5

In ambient computing – one term for contextually aware computing – massive amounts of data from multiple sensors enable a computer to adapt to user and situation, rather than the other way around. In the audio world, for example, consider the ever present lag in the time it takes to transcribe or translate live audio from one language to another during video conferencing, news broadcasting and similar scenarios. People have accepted this inconvenience for years, but ambient computing enabled by advanced data parallel processing capabilities of GPUs creates the potential for audio interpretation and voice translation to other languages immediately, in real time.

This more contextually aware computing also opens the possibility for capabilities like speech recognition in audio recording application.This more contextually aware computing also opens the possibility for capabilities like speech recognition in audio recording applications – so that, for example, software will automatically determine who is speaking in a conference room and identify the different speakers in the auto-generated transcript of the conference. Another potential application is for software that will dynamically gauge the acoustics in a room and tweak output accordingly for whatever mobile device you're using.

These kinds of audio applications translate into digital security as well – in, for example, systems that grant user access to physical locations and computer systems by recognizing and authenticating users based on voice. And while we're on the subject of security, GPU capabilities could enable targeted malicious software scans that eliminate the painfully slow runtimes associated with today's anti-virus programs.

These aren't blue-sky speculations. Many of the capabilities exist today in labs all over the world. Why haven't they found their way to the mainstream yet? The answer is in the limitations of existing hardware architecture and software programming models. And this is why heterogeneous systems architecture is vital to enabling the next era of computing innovation.

REFERENCE

[1]. www.openacc.org/sites/default/files/OpenACC%202%200.pdf
[2]. en.wikipedia.org/wiki/Heterogeneous_computing

[3]. http://www.drdobbs.com/parallel/the-openacc-execution-model/240006334?pgno=1
[4]. http://www.diva-portal.org/smash/get/diva2:655634/FULLTEXT01.pdf
[5]. http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-system-architecture-hsa/
[6]. www.youtube.com/watch?v=r6r2NJxj3kI
[7]. http://www.ece.cmu.edu/~ece447/s13/lib/exe/fetch.php?media=onur-447-spring13-lecture33-heterogeneousmulticore-afterlecture.pdf
[8]. www.nvidia.com/gpudirectives
[9]. ww.linksceem.eu/ls2/images/stories/OpenACC.pdf
[10]. www.training.praceri.eu/uploads/tx.../HeterogeneousComputingJU.pdf
[11]. www.pgroup.com/lit/whitepapers/pgi_accel_prog_model_1.2.pdf

