Cloud X: A Platform as a Service for CT Reconstruction Research and Development

Eric Papenhausen, Ziyi Zheng, and Klaus Mueller

***Abstract* – Many CT reconstruction algorithms, especially in iterative CT, are constructed from a few common algorithms (e.g. backprojection and forward projection). These algorithms often dominate the computation cost during CT reconstruction. GPUs can provide an order of magnitude speed-up over conventional CPU implementations. However, without the proper hardware or willingness to develop the specialized software, these speedups will remain unexploited. In addition to implementing these common algorithms, sharing research can also be a burden. Assuming one is able to get the software to run someone else's experiments, one can spend an entire work day tracking down the appropriate dependencies and modifying hard coded file paths to run it on a different machine. These obstacles hinder productivity and slow research. In this paper, we present a cloud computing framework that aims to make research in the medical imaging domain more efficient by providing a number of common GPU accelerated algorithms and allows for efficient sharing of research through a virtual workspace. It also has an interface that allows users to present and analyze their research.**

***Index Terms*—Cloud Computing, GPU, CT reconstruction**

I. INTRODUCTION

With the introduction and rapid adoption of GPUs to the medical imaging domain [6][8][9][13], it is becoming increasingly important to leverage the processing power of GPUs to make the leap from research to clinical use. Developing software on GPUs, however, can seem like a daunting task. It not only requires the appropriate hardware and software tools, but a shift in the mindset of how a program operates. This can be especially difficult for theoretical researchers, who have to rely on someone else to accelerate their algorithms on graphics hardware.

Furthermore, comparing different CT reconstruction methods can require a re-implementation of a previous technique. A popular iterative CT reconstruction algorithm is ASD-POCS [10]. When a new iterative CT algorithm is proposed, it is often compared to ASD-POCS and other techniques [1][2][3] to demonstrate the validity of the new algorithm. Researchers will often have to re-implement multiple reconstruction algorithms to compare to their new method; then mention that they are unsure of their implementation of the old methods.

Most research today is relatively one sided (i.e. the authors explain how an experiment was set up and then present results), but there is no easy way for the reader to validate and further explore a presented algorithm. There are a number of parameters that can affect the quality of a CT reconstruction algorithm (i.e. number of projections, dose per projection, anatomy and pathology, etc.), but only a subset of these parameters are typically presented.

In this paper, we present a cloud computing framework that will be able to solve these problems by providing three main services. First, it will provide a virtual workspace which will facilitate the sharing of research and contain a number of pre-loaded GPU accelerated CT reconstruction algorithms. Second, it will allow remote execution of GPU accelerated algorithms, thus allowing users to take advantage of graphics cards without having to purchase the hardware or develop GPU accelerated applications. Finally, it will provide an interface that will allow authors to post their research and allow users to replicate experiments and explore the parameter space through their web browser.

We begin in section II by presenting an interface that we have developed to allow users to explore the parameter space of an iterative CT reconstruction algorithm through their web browser. The rest of the paper will then be focused on how to build a scalable framework around this interface and the other services that this framework will provide. In section III, we will discuss the current technology that will allow us to build this cloud computing framework. Section IV will describe the architecture of our framework and introduce the concept of the virtual workspace. In section V, we revisit our web interface and describe how it will operate in the cloud. Section VI presents a library that will allow remote execution of GPU accelerated algorithms. Finally, section VII concludes the paper.

II. WEB INTERFACE

We have developed a prototype of a web interface that allows users to explore the parameter space of an iterative CT reconstruction algorithm. More specifically the algorithm is OS-SIRT [11] using a bilateral filter (BLF)[12] for regularization. The bilateral filter is defined in the following equations, where x is the location of smoothing, and t is an offset. The BLF is essentially a Gaussian that falls off as a function of both spatial and value deviation.

Eric Papenhausen, Ziyi Zheng (now with Amazon.com, Seattle), and Klaus Mueller are (or have been) with the Visual Analytics and Imaging Lab, Computer Science Department, Stony Brook University, Stony Brook, NY 11777 USA (phone: 631-632-1524; e-mail: {epapenhausen, zizhen, mueller}@cs.sunysb.edu).

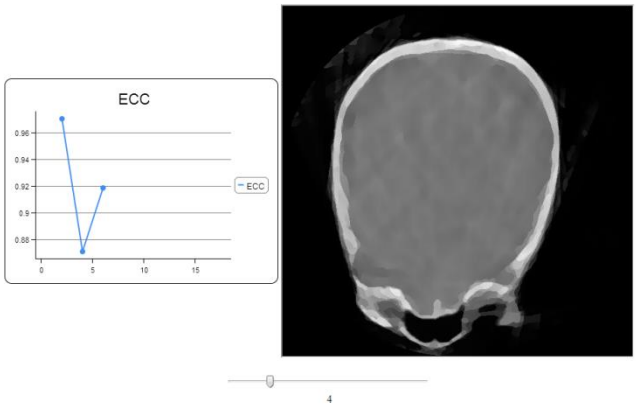

Figure 1. A view of the web interface that we have developed.

$$\alpha_{BLF}(x,t,f) = c_d(t)s_r(f(x),f(x+t))$$
$$c_d(t) = G_{\sigma_d}(\|t\|) \quad (1)$$
$$s_r(f(x),f(x+t)) = G_{\sigma_r}(|f(x)-f(x+t)|)$$

Where $G_\sigma(x)$ is the Gaussian kernel:

$$G_{\sigma_r}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{x^2}{2\sigma^2}) \quad (2)$$

We allow the user to explore the filter's parameter space interactively using a slider to determine the amount of smoothing. The slider changes the parameters of the BLF to control the smoothing such that the value chosen by the slider determines σ_r and σ_d (i.e. $\sigma_r = \sigma_d$).

We implemented this interface using HTML and JavaScript on the front end, PHP as the server side scripting language, and finally using C and CUDA to handle the CT reconstruction. The act of moving the slider triggers a request to the web server. The web server then writes the parameter value, chosen by the slider, to a file. A simple C program waits for this file to be updated, and when it is, it calls the OS-SIRT function with the appropriate parameters, and waits for the volume to be reconstructed. The central slice of the reconstructed volume along with its E-CC score is then passed back to the webserver to be displayed to the user. Fig. 1 shows the web interface that the user sees. The graph on the left plots the quality metric curve evolved with each slider update and the right shows the current image.

Currently, this setup is running on a single desktop machine with a NVIDIA GeForce GTX 480. It takes around 11 seconds for each reconstruction (i.e. 512^3 volume for OS-SIRT 10 with 60 projections and a BLF for regularization). We have available to us, however, a GPU cluster that contains 8 NVIDIA Tesla M2050 GPUs. The framework that we present in the rest of this paper is designed to run on this GPU cluster. We expect the reconstruction time to drop to between 1.3 and 1.8 seconds. This, however, does not mean that the user will have to wait 1.3-1.8 seconds every time he moves the slider, since we will include a number of optimization strategies, presented throughout this paper.

III. CLOUD COMPUTING

With the popularization of cloud computing, a number of open source technologies have been developed to support the infrastructure and system level requirements that are required by cloud computing frameworks [4][7]. Openstack in particular is a popular cloud operating system. It contains a number of networking and storage solutions that are required by the typical cloud computing infrastructure. It also allows for the creation and management of multiple virtual machines (i.e. software that emulates the computer architecture of a real machine). This is particularly important for efficient sharing of computational resources. Unfortunately, openstack currently does not contain any resources for efficient GPU virtualization (i.e. sharing of GPU resources).

Gvirtus is an open source GPU virtualization technology. It allows multiple virtual machines to share the computational resources provided by one or more GPUs. To a user operating on a virtual machine, however, it appears as if there is one dedicated GPU. This abstraction allows for efficient GPU sharing, while still providing the user an interface he is familiar with.

Another important tool for cloud computing is the network file system[5] (NFS). This allows many users to connect to a single file system concurrently. To the user, however, the NFS looks like a regular file system. This allows for efficient sharing of information.

IV. FRAMEWORK ARCHITECTURE

The cloud computing framework we are currently developing and have initial results for utilizes all of the technology presented in section III (i.e. openstack, gvirtus, and NFS) as well as custom improvements to make the overall system more efficient. The virtual workspace is designed as a "platform as a service" (PaaS) and will allow users to work in the cloud through their web browser. Each user will have a separate virtual machine assigned to them and will contain a number of relevant GPU accelerated CT reconstruction algorithms pre-configured on their workspace. There will be one virtual machine assigned to handle the remote execution of GPU accelerated CT reconstruction algorithms. Finally, there will be one virtual machine to function as a web server, and will be configured to allow users to upload their implementations, allowing it to be executed through a web browser. Since every service is being handled through virtual machines, the 8 GPUs in our cluster can be efficiently shared using gvirtus.

Gvirtus consists of two segments, a front end, located on the virtual machine, and a back end, located on the host operating system. The front end provides a library that acts as a wrapper to many CUDA functions. The front end redirects calls to these functions to the back end. The back end executes the CUDA operations and returns the results back to the front end on the virtual machine.

The network file system will contain a number of commonly used dependencies (i.e. CUDA libraries, include files, etc.) as well as a number of datasets available for CT reconstruction. It will also include a public folder for each user. The public folder allows a user to share his research with the rest of the community. Other users can easily download software from a public folder into their workspace. Since the dependencies are in a shared file system, the software will run without requiring additional modifications.

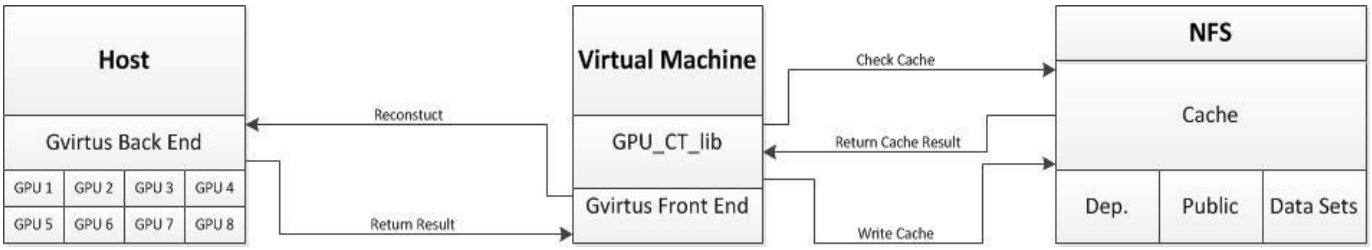

Figure 2. An illustration of the framework presented in this paper. When a function from the "GPU_CT_lib" is called, the system first checks the NFS to see if the result lies in cache. If so, the result is returned. On a cache miss, the appropriate reconstruction function is redirected to the back end (i.e. host machine) through Gvirtus; the result is reconstructed and returned. That result is then written to the cache on the NFS.

The network file system will also contain a block of space to act as cache. A feature vector, containing all the parameters of a previously reconstructed volume (i.e. dataset, number of projections, etc.) and the volume itself, will be recorded. With this strategy, we can alleviate GPU resource contention by only executing CT reconstruction algorithms for which we do not already have the results. Figure 2 shows how this strategy interacts with the rest of the system. We will, however, provide a "no-cache" option for situations where obtaining accurate timings of an algorithm are important.

V. WEB INTERFACE REVISITED

A critical aspect of performing research is presenting the results. This is usually fulfilled by presenting an image of a slice of a reconstructed volume followed by a discussion of how it compares to a gold standard. The presented slice, however, is only one of many possible reconstructions. Results can vary greatly depending on the parameters that are chosen for the reconstruction. Moreover, the quality of the algorithm can differ depending on the anatomy of the dataset that is being reconstructed. By providing an interface that allows readers to access the research in question, experiments can be easily replicated and the algorithm can be thoroughly evaluated.

In section II, we described the details of a prototype interface that addressed these problems. This, however, was specific to OS-SIRT and the BLF. To make this interface more useful, there needs to be a mechanism that allows researchers to attach their project to this interface.

In order for researchers to utilize this interface, however, the process of uploading algorithms needs to be simple. The researcher will need to implement certain methods that will be called when a parameter is changed via a slider. These methods will provide the new parameter settings as an argument. The researcher will simply set the parameters of his algorithm based on the values provided by the web interface, and execute his algorithm. Figure 3 shows a pseudo-code example of the functions that will be needed by the web interface.

```
void sliderSetParam1(float val) { sigma = val; }

void sliderSetParam2(float val) { lambda = val; }

void execute() { . . .}
```

Figure 3. A pseudo-code example demonstrating the functions that will need to be defined for the web interface. The functions with the "slider" prefix will simply set the parameters of the algorithm. The execute function will actually perform the algorithm

Once the appropriate functions are implemented, their program will need to be compiled as a dynamic library. The compiled code will be placed in a directory that is accessible by the web server. By compiling the project as a dynamic library, the web server can call the functions it needs to, without having to restart the server itself. This is important because multiple users will be sharing the same web server. When the dynamic library is placed in the appropriate directory, users will be able to visit a web page specific to the project in question, and will be able to explore and execute the algorithm in real time.

There are some performance issues since there will still be a noticeable delay from when the slider is changed, to when the reconstruction is complete. In section II, we anticipated that the reconstruction time would be between 1.3-1.8 seconds using our cluster of 8 GPUs. With multiple users, this can take much longer since a user will have to wait for a GPU to be free before reconstruction can take place. Some extra optimization strategies are required to reduce GPU resource contention and ensure that web interface remains interactive.

One strategy that we have already presented is the use of cache. Instead of immediately reconstructing the volume, we check to make sure that it has not already been reconstructed. If it has, we simply return the results without performing any extra computation. This is particularly useful with the slider, as users will often move the slider back and forth, reviewing parameter settings they have already seen.

Another strategy is to only reconstruct a subset of the volume. Since only the central slice is displayed to the user, most of the three-dimensional volume is never seen. We estimate that we can reconstruct a 512x512x8 subset of the volume in 0.3-0.7 seconds using our 8 GPU cluster. This strategy combined with caching will greatly reduce the strain on the GPU server while still providing the user with a real-time interface in which he can explore the parameter space of a CT reconstruction algorithm.

VI. REMOTE EXECUTION

The virtual workspace provides a number of resources that are beneficial when starting a new project. This option may not be ideal, however, when maintaining an existing project. A researcher may still take advantage of the processing power of GPUs by using our remote execution library. This library aims to provide a simple method for executing a number of commonly required GPU accelerated algorithms (i.e. backprojection, forward projection, etc.) in the cloud.

The remote execution library will provide a number of wrapper methods to the GPU accelerated algorithms that are provided in the virtual workspaces. These methods operate by acting as a liaison between the user's machine, and the virtual machine in the cloud responsible for processing remote execution requests.

```
cloudTCPConnect();
cloudAllocMem(vol, size);
cloudAllocMem(volp1, size);
cloudAllocMem(proj, projection_size);
for each iteration
    cloudProject_GPU(vol, proj);
    cloudCorrect_GPU(proj, "dataset");
    cloudBackproject_GPU(volp1, proj);
    cloudBLF_GPU(volp1);
    cloudVoxelUpdate_GPU(vol, volp1, lambda);
end for
cloudTCPDestroy();
```

Figure 4. An iterative CT implementation using the remote execution library. Functions with the prefix cloud will be executed on the cloud.

Figure 4 shows a pseudo-code example of how users can interact with the remote execution library. On the server side, a virtual machine will be tasked with listening and waiting for a connection request from a client. On the client side, a TCP connection is initiated with the server. This allows for effective communication between the client and the server. A call to execute a GPU accelerated algorithm by the client sends a signal to the server, indicating the parameters the user requested (i.e. algorithm, dataset, etc.). The server then executes the requested function in a similar manner illustrated in Figure 2. The client can then request the results to be transferred from the cloud to the user's machine.

We recognize that most CT reconstruction pipelines (e.g. iterative CT), however, can be highly integrated (i.e. the results of the "project" section are given as input in the "backproject" section). To reduce the number of memory copies between the client and server, we provide a mechanism to allow the user to allocate space on the cloud. With this functionality, the user can allocate an array to store the results of the "project" function, and pass it as input to the "backproject" function. This reduces costly data transfers between the client and server.

If, however, one component of this pipeline is missing in our library then it becomes useless; since any performance gained by exploiting GPUs will be wiped out by the cost of transferring data to and from the server on every iteration. To avoid this situation, we let users extend the remote execution library by allowing them to upload custom implementations to the remote execution server. A simple addition to the wrapper library, is then required. In this fashion, users can easily expand the library to suit their needs.

With the remote execution library, users can exploit the massive speed-ups GPUs provide without the burden of developing highly specialized software. This library will be primarily useful for those looking to quickly replace sequential CPU implemented CT algorithms with GPU accelerated implementations.

VII. CONCLUSIONS

In this paper, we presented an interface for dynamically exploring the parameter space of an iterative CT reconstruction algorithm through a web browser. We then explained how we can build a scalable framework to allow many users to interact with this interface in a cloud environment while handling practical issues related to resource contention. The technology developed for cloud based computing infrastructures are quite powerful and allow us to do so much more than simply host web interface for exploring iterative CT reconstruction parameters.

This cloud framework can be used to provide resources that are not necessarily available (i.e. hardware and software) to many researchers. We address many practical issues in the medical imaging community relating to the sharing and presentation of research. The cloud computing framework presented in this paper will be a powerful force for good in the medical imaging community.

REFERENCES

[1] A. Andersen and A. Kak, "Simultaneous algebraic reconstruction technique (SART): A superior implementation of the ART algorithm," Ultrason. Imaging, 6:81–94, 1984

[2] A. Chambolle, "An Algorithm for Total Variation Minimizations and Applications," J. Mathematical. Imaging and Vision 20(1-2) 89-97 (2004).

[3] P. Gilbert, "Iterative methods for the 3D reconstruction of an object from projections," Journal of Theoretical Biology 76, 105–117 (1972).

[4] G. Giunta, R. Montella, G. Agrillo, and G. Coviello. "A GPGPU transparent virtualization component for high performance computing clouds". In Euro-Par 2010 - Parallel Processing, volume 6271 of Lecture Notes in Computer Science, chapter 37, pages 379–391–391. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010.

[5] A. Muthitacharoen, B. Chen, and D. Mazieres. A Low-bandwidth Network File System. In Proc. 18th SOSP, Oct. 2001.

[6] Y. Okitsu, F. Ino and K. Hagihara. "High-Performance Cone Beam Reconstruction Using CUDA Compatible GPUs," Parallel Computing,36(2-3):129-141, 2010.

[7] OpenStack LLC, "OpenStack: The Open Source Cloud Operating System," 21-Jul-2012. [Online]. Available: http://www.openstack.org/software/.

[8] E. Papenhausen, Z. Zheng, and K. Mueller, "GPU-Accelerated Back-Projecting Revisited: Squeezing Performance by Careful Tuning," Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine (Potsdam, Germany, 2011)

[9] H. Scherl, B. Keck, M. Kowarschik, J. Hornegger, "Fast GPU-based CT reconstruction using the Common Unified Device Architecture(CUDA)," IEEE Medical Imaging Conference, 6: 4464-4466, Honolulu, HI, 2007.

[10] E.Y. Sidky and X. Pan, "Image Reconstruction in Circular Cone-Bean Computed Tomography by Constrained, Total Variation Minimization", Phys. Med. Biol. 53 (2008) 4777-4907

[11] F. Xu, W. Xu, M. Jones, B. Keszthelyi, J. Sedat, D. Agard and K. Mueller, "On the Efficiency of Iterative Ordered Subset Reconstruction Algorithms for Acceleration on GPUs," Computer Methods and Programs in Biomedicine 98(3) 261-270 (2010).

[12] W. Xu, K. Mueller, "Evaluating Popular Non-Linear Image Processing Filters for their Use in Regularized Iterative CT," IEEE Medical Imaging Conference, Knoxville, TN, October, 2010

[13] Z. Zheng, K. Mueller "Cache-Aware GPU Memory Scheduling Scheme for CT Back-Projection," IEEE Medical Imaging Conference, Oct. 2010.