TABLE OF CONTENTS

ABSTRACT

Many numerical methods based on Radial Basis Functions (RBFs) are gaining popularity in the geosciences due to their competitive accuracy, functionality on unstructured meshes, and natural extension into higher dimensions. One method in particular, the Radial Basis Function-generated Finite Differences (RBF-FD), has drawn significant attention due to its comparatively low computational complexity versus other RBF methods, high-order accuracy (6th to 10th order is common), and embarrassingly parallel nature.

Similar to classical Finite Differences, RBF-FD computes weighted differences of stencil node values to approximate derivatives at stencil centers. The method differs from classical FD in that the test functions used to calculate the differentiation weights are n-dimensional RBFs rather than one-dimensional polynomials. This allows for generalization to n-dimensional space on completely scattered node layouts.

We present our effort to capitalize on the parallelism within RBF-FD for geophysical flow. Many HPC systems around the world are transitioning toward significantly more Graphics Processing Unit (GPU) accelerators than CPUs. In addition to spanning multiple compute nodes, modern code design should include a second level of parallelism targeting the many-core GPU architectures. We will discuss our parallelization strategies to span computation across GPU clusters, and demonstrate the efficiency and accuracy of our parallel explicit and implicit solutions.

CHAPTER 1

INTRODUCTION

Background on GPUs. GPUs were introduced in 80's see master's thesis.

Originally GPUs were designed as parallel rasterizing units. They had limited logic control in contrast to the serial CPUs and their advanced branching and looping logic.

Gradually new and complex logic was added to the GPU to produce the shader languages that allowed developers to customize specific parts of the rendering pipeline. This allowed scientific problems such as the diffusion equation cite Lore and others to be solved in process of rendering. In other words, the GPU was tricked into computing.

The year 2006 brought the modern age of GPU computing with the introduction of CUDA from NVidia. The high level language allowed scientists to leverage the GPU as a parallel accelerator without all of the overhead of setting up graphics contexts and tricking the hardware into computing. Memory management is still the developer's responsibility, but compiler transforms generic C/Fortran code to GPU instruction set.

Scientific Computing has seen a widespread adoption of GPGPU because of the goal to get to "exa-scale" computing, which may only be possible in the near future with the help of GPU accelerators [3].

NVidia is not the only company involved in many core parallel accelerators. Other groups like AMD and Intel have been increasing the number of cores as well. The end effect is a hybridization where CPUs look similar to GPUs and vice-versa.

Until 2009, the hardware distinction required that developers target parallelism on CPUs and GPUs using different languages. Then the OpenCL standard was drafted and implemented. OpenCL is a parallel language that strives to provide functional portability rather than performance.

We focus on the OpenCL language within this dissertation with confidence that hardware will change frequently. In fact, every 18 months cite shows a new release of GPU hardware, manycore CPU hardware and extensions to parallel languages. But if hardware is constantly changing, then we need to focus on a high level implementation that allows portability. We need a language like OpenCL to carry our implementations into the future regardless of what hardware and which company survive.

Scientific problems and the need for computational methods. Many scientific problems of importance can be expressed as a collection of partial differential equations defined for some domain. In order to solve these problems , computational numerical methods are employed on a discretized version of the domain. Traditionally, three major

categories exist for PDE solutions: finite difference (FD), finite element (FEM) and spectral element (SEM) [**?**]. Interestingly, all three of these methods rely on an underlying mesh, making them *meshed methods*. While each has had a turn in the spotlight, more recently a new category, or rather a generalization on all three previous categories, has emerged: *meshfree methods*.

Traditional methods (FD, FV, FEM, SEM) and their dependence on structured meshes and "nice" connections (e.g., Delaunay).. Note, finite element allows for triangle meshes, but well balanced Delaunay are preferred to limit ill conditioning. The same regularity in nodes is preferred with RBF methods. The reason for this is to keep best conditioning of the system.

Leaders in computational science are cobbled together with metaphorical bubble gum and duct tape. Many scientists neglect to plan in advance for items like:

- generic point clouds; most methods require a point cloud include some connectivity information for structured and unstructured mesh.
- we need to add adaptive mesh refinement
- we need to avoid pole singularities
- save on interpolations and differences but get similar answer in O(nm).

recently, meshfree methods surfaced leveraging RBFs.

RBFs have interesting history. Started with interpolation, but went to collocation with global SEM scheme. The global scheme has $O(N^2)$ complexity..

Most recently RBFs went to RBF-FD. Introduces sparsity and reduces cost of scaling..

For RBF problems in general there is limited work that scales the methods to large problems..

Large problems require high performance computing. Given lack of scaling in literature, it has been irrelevant until recently to assess the possibility of solving PDEs with RBF-FD on GPUs..

what is new in the thesis (summarize: 1-2 pages). The large scale contributions of this thesis can be summarized as follows:

- This thesis covers details related to the application of the RBF-FD method for both explicit and implicit solutions for PDES.
- Our domain decomposition algorithm provides the first known parallelization of RBF-FD across multiple CPUs.
- Furthermore, we offload intense computational tasks to the GPU creating the first ever single- and multi-GPU implementation of RBF-FD.

As part of the research on RBF-FD within those three areas we additionally contributed the following:

- Application of RBF-FD to Centroidal Voronoi Tessellation grids.
- Approximate Nearest Neighbor methods for faster neighbor queries and improved system conditioning
- Multi-GPU implementations of sparse iterative Krylov solvers

Research Statement. What motivated this research?

The goal of this dissertation is to present a unified approach to parallel solutions of Partial Differential Equations (PDEs) with a method called Radial Basis Function-generated Finite Differences (RBF-FD).

fix: Many scientific problems of importance can be expressed as a collection of PDEs. Solutions to these problems provide answers to many simple questions such as the current temperature of a material, or perhaps the current position of a moving object. Complex and coupled PDEs can simulate the growth of zebra stripes or cheetah spots [30] or even model the flow of fluids. Need refs for examples

In order to solve these problems, computational numerical methods are employed on a discretized version of the domain. Traditionally, three (and how would I classify FV? PartOfUnity?) major categories exist for PDE solutions: finite difference (FD), finite element (FEM) and spectral element (SEM) [18]. Interestingly, all three of these methods rely on an underlying mesh, making them *meshed methods*. While each has had a turn in the spotlight, more recently a new category, or rather a generalization on all three previous categories, has emerged: *meshfree methods*.

The first task in traditional meshed methods is to generate an underlying grid/mesh. Node placement can be done in a variety of ways including uniform, non-uniform and random (monte carlo) sampling, or through iterative methods like Lloyd's algorithm that generate a regularized sampling of the domain (see e.g., [17]). meshed methods have constraints on edge length and angle. Delaunay answers this, but is costly to compute Mesh2d, Triangle, DIstmesh In addition to choosing nodes, meshed methods require connectivity/adjacency lists to form stencils (FD) or elements (FEM, SEM)—this implies an added challenge to cover the domain closure with a chosen element type. While these tasks may be straightforward in one- or two-dimensions, the extension into higher dimensions becomes increasingly more cumbersome [37].

Complex geometries, irregular boundaries and mesh refinement also pose a problem for meshed methods. As the complexity of the geometry/boundaries increases, so too should the resolution of the approximating mesh in order to accurately reconstruct the detail present. A naïve approach to refinement increases the density of nodes uniformly across the domain, adding much more computation and memory storage than necessary for activity that is localized to sub-regions of the domain. Multiresolution methods attempt to compromise between accurate approximation of the domain and reduced resolution by one of two approaches: a) *multilevel methods* that decompose the model into a hierarchy with several levels of mesh detail, then only use a level when it is required to capture phenomena; and b) *adaptive irregular sampling* which has one level of detail, but non-uniform nodal density concentrated in areas of high activity [32]. Such techniques require robust methods and complex code capable of either coarsening/smoothing the approximate solutions to new level, or handling non-uniform node placement, element size etc.

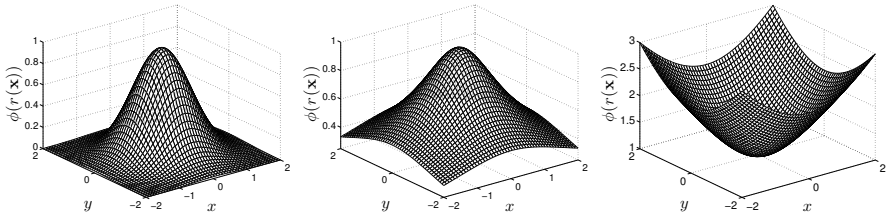
Ideally, we seek a method defined on arbitrary geometries, that behaves regularly in any dimension, and avoids the cost of mesh generation. The ability to locally refine areas of interest in a practical fashion is also desirable. Fortunately, meshfree methods provide all of these properties: based wholly on a set of independent points in n-dimensional space, there is minimal cost for mesh generation, and refinement is as simple as adding new points where they are needed.

Since their adoption by the mathematics community in the 1980s ([18]), a plethora of meshfree methods have arisen for the solution of PDEs. For example, smoothed particle hydrodynamics, partition of unity method, element-free Galerkin method and others have been considered for fluid flow problems [12]. For a recent survey of methods see [37].

A subset of meshfree methods of particular interest to the community today revolves around Radial Basis Functions (RBFs). RBFs are a class of radially symmetric functions (i.e., symmetric about a point, x_j, called the *center*) of the form:

$$\phi_j(\mathbf{x}) = \phi(r(\mathbf{x})) \tag{1.1}$$

where the value of the univariate function ϕ is a function of the Euclidean distance from the center point $\mathbf{x}_j$ given by $r(\mathbf{x}) = ||\mathbf{x} - \mathbf{x}_j||_2 = \sqrt{(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2}$. Examples of commonly used RBFs are available in Table **??** with their corresponding plots in Figure 1.1. RBF methods are based on a superposition of translates of these radially symmetric functions, providing a linearly independent but non-orthogonal basis used to interpolate between nodes in n-dimensional space. An example of RBF interpolation in 2D using 15 Gaussians is shown in Figure 2.1, where $\phi_j(r(\mathbf{x}))$ is an RBF centered at $\{\mathbf{x}_j\}_{j=1}^n$.



(a) Gaussian (b) Inverse Multiquadric (c) Multiquadric

Figure 1.1: Commonly used RBFs.

With a history extending back four decades for RBF interpolation schemes [31], and two decades for RBFs applied to solving PDEs [33], many avenues of research remain untouched within their realm. Being a meshless method, RBF methods excel at solving problems that require geometric flexibility with scattered node layouts in n-dimensional space. They naturally extend into higher dimensions without significant increase in programming complexity [22, 53]. In addition to competitive accuracy and convergence compared with other state-of-the-art methods [22, 23, 20, 53, 19], they also boast stability for large time steps.

Like most numerical methods, RBFs come with certain limitations. For example, RBF interpolation is—in general—not a well-posed problem, so it requires careful choice of positive definite or conditionally positive definite basis functions [32, 18]. The example 2D RBFs presented in Figure 1.1 are infinitely smooth and satisfy the (conditional) positive definite requirements.

Infinitely smooth RBFs depend on a shape or support parameter ϵ that controls the width of the function. The functional form of the shape function becomes $\phi(\epsilon r)$. Decreasing ϵ increases the support of the RBF and in most cases, the accuracy of the interpolation, but worsens the conditioning of the RBF interpolation problem [42]. The conditioning of the system also dramatically decreases as the number of nodes in the problem increases. Fortunately, recent algorithms such as Contour-Padé [24] and RBF-QR [28, 26] allow for numerically stable computation of interpolants in the nearly flat RBF regime (i.e., $\epsilon \to 0$) where high accuracy has been observed [36, 29].

Historically, the most common way to leverage RBFs for PDE solutions is in a global interpolation sense. That is, the value of a function value or any of its derivatives at a node location is a linear combination of all the function values over the *entire* domain, just as in a pseudospectral method. If using infinitely smooth RBFs, this leads to spectral (exponential) convergence of the RBF interpolant for smooth data [25]. As discussed in [23], global RBF methods require $O(N^3)$ floating point operations (FLOPs) in pre-processing, where N is the total number of nodes, to assemble and solve a dense linear system for differentiation coefficients. The coefficients in turn are assembled into a dense Differentiation Matrix (DM) that is applied via matrix-vector multiply to compute derivatives at all N nodes for a cost of $O(N^2)$ operations. assumes explicit scheme

Alternatively, one can use RBF-generated finite differences (RBF-FD) to introduce sparse DMs (Note: for pure interpolation, compactly supported RBFs can also introduce sparse matrices [51]). RBF-FD was first introduced by Tolstykh in 2000 [48], but it was the simultaneous, yet independent, efforts in [45], [47], [52] and [11] that gave the method its real start. RBF-FD share advantages with global RBF methods, like the ability to function without an underlying mesh, easily extend to higher dimensions and afford large time steps; however spectral accuracy is lost. Some of the advantages of RBF-FD include high computational speed together with high-order accuracy (6th to 10th order accuracy is common) and the opportunity for parallelization.

The RBF-FD method is similar in concept to classical finite-differences (FD): both methods approximate derivatives as a weighted sum of values at nodes within a nearby neighborhood. The two methods differ in that the underlying differentiation weights are exact for RBFs rather than polynomials.

As in FD, increasing the stencil size n increases the accuracy of the approximation. Given N total nodes in the domain (such as on the surface of a sphere), N linear systems, each of size $n \times n$, are solved to calculate the differentiation weights. Since $n \ll N$, the RBF-FD preprocessing complexity is dominated by $O(N)$—much lower than for the global RBF method of $O(N^3)$—with derivative evaluations on the order of $O(nN) \implies O(N)$ FLOPs.

CHAPTER 2

RELATED WORK

2.1 GPUS for explicit and implicit methods

2.2 GPUS/RBF

[43, 44]. Our paper [8] introduced the first implementation of RBF-FD to span multiple CPUs and multiple GPUs.

2.3 RBF/Finite-Difference

2.3.1 Global RBF Method

2.3.2 RBF-FD

RBF-FD have been successfully employed for a variety of problems including Hamilton-Jacobi equations [11], convection-diffusion problems [12, 46], incompressible Navier-Stokes equations [45, 13], transport on the sphere [27], and the shallow water equations [21].

2.4 Preconditioners

Many preconditioners are based on physical properties of the domain, PDE and the spectral properties of the numerical method.

2.5 Sparse matrix libraries

2.6 Various GPU/based libraries

2.6.1 CUSP

GMRES in CUSP is based on the Givens rotations for the Arnoldi process.

2.6.2 ViennaCL

GMRES in ViennaCL is based on Householder reflections for the Arnoldi process.
We opted to leverage the ViennaCL package in most of our implementations.

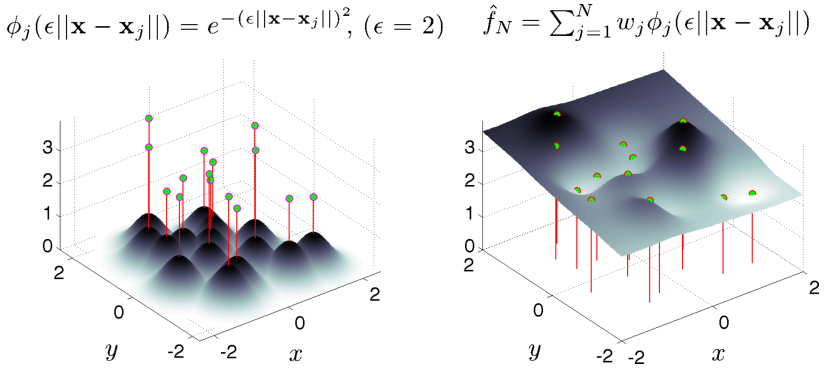


Figure 2.1: RBF interpolation using 15 translates of the Gaussian RBF with $\epsilon = 2$. One RBF is centered at each node in the domain. Linear combinations of these produce an interpolant over the domain passing through known function values.

2.7 Parallel implementations of RBF

As N grows larger, it behooves us to work on parallel architectures, be it CPUs or GPUs. With regard to the latter, there is some research on leveraging RBFs on GPUs in the fields of visualization [15, 50], surface reconstruction [14, 10], and neural networks [9]. However, research on the parallelization of RBF algorithms to solve PDEs on multiple CPU/GPU architectures is essentially non-existent. We have found three studies that have addressed this topic, none of which implement RBF-FD but rather take the avenue of domain decomposition for global RBFs (similar to a spectral element approach). In [16], Divo and Kassab introduce subdomains with artificial boundaries that are processed independently. Their implementation was designed for a 36 node cluster, but benchmarks and scalability tests are not provided. Kosec and Šarler [35] parallelize coupled heat transfer and fluid flow models using OpenMP on a single workstation with one dual-core processor. They achieved a speedup factor of 1.85x over serial execution, although there were no results from scaling tests. Yokota, Barba and Knepley [54] apply a restrictive additive Schwarz domain decomposition to parallelize global RBF interpolation of more then 50 million nodes on 1024 CPU processors. Only Schmidt et al. [44] have accelerated a global RBF method for PDEs on the GPU. Their MATLAB implementation applies global RBFs to solve the linearized shallow water equations utilizing the AccelerEyes Jacket [2] library to target a single GPU.

2.8 Contributions of this Work

Within this dissertation, we have developed the first implementation of RBF-FD to span multiple CPUs. Each CPU has a corresponding GPU attached to it in a one-to-one correspondence. We thus also introduced the first known implementation of accelerated RBF-FD on the GPU.

continue with outline of chapters We present our explicit and implicit solutions to PDEs with our multi-GPU RBF-FD implementation.

CHAPTER 3

FRAGMENTS (INTEGRATE ABOVE)

with the goal of solving coupled fluid flows modelling the physics (?) in the interior of the earth. We require both advective terms and diffusive terms. The components of a good pressure fluid solver are explicit and implicit differentiation. Whether we solve a steady-state PDE implicitly or a hyperbolic PDE with an implicit time-scheme, we require an *implicit solver*. An implicit solver is nothing but a method of assembling a differentiation matrix, forcing terms on the RHS and then solving the linear system. A direct LU factorization would suffice if our system is dense, but with RBF-FD the system is sparse. Sparse systems can be efficiently solved under the right conditions through sparse iterative solvers. :

Sufficient understanding of RBF interpolation led to the development of the first global RBF method for PDEs in [33]. Most popular among global methods is collocation, wherein RBF interpolation approximates the PDE solution by solving a large, dense linear system. In some cases RBF collocation demonstrates higher accuracy for the same number of nodes when compared to other state-of-the-art pseudospectral methods (e.g., [36] [**?**] [**?**]). In [**?**], spectral accuracy was demonstrated for hyperbolic PDEs even with local refinement of nodes in time.

Unfortunately—ill-conditioning aside—collocation methods are prohibitively expensive to solve when scaled to a large number of nodes. Assuming the collocation matrix does not change in time, global methods, with their dense systems, scale at $O(N^3)$ operations for initial preconditioning/preprocessing followed by $O(N^2)$ operations every time-step. This complexity is consistent with any collocation scheme. However, by introducing sparsity into the system (e.g., using compactly supported RBFs), the complexity is somewhat reduced.

General Purpose GPU (GPGPU) computing is one of today's hottest trends within scientific computing. The release of NVidia's CUDA at the end of 2006 marked both a redesign of GPU architecture, plus the addition of a new software layer that finally made GPGPU accessible to the general public. The CUDA API includes routines for memory control, interoperability with graphics contexts (i.e., OpenGL programs), and provides GPU implementation subsets of BLAS and FFTW libraries [38]. After the undeniable success of CUDA for C, new projects emerged to encourage GPU programming in languages like FORTRAN (see e.g., HMPP [1] and Portland Group Inc.'s CUDA-FORTRAN [39]).

In early 2009, the Khronos Group–the group responsible for maintaining OpenGL–announced a new specification for a general parallel programming lanugage referred to as the Open Compute Language (OpenCL) [34]. Similar in design to the CUDA language—in

many ways it is a simple refactoring of the predecessor—the goal of OpenCL is to provide a mid-to-low level API and language to control any multi- or many-core processor in a uniform fashion. Today, OpenCL drivers exist for a variety of hardware including NVidia GPUs, AMD/ATI CPUs and GPUs, and Intel CPUs.

This *functional portability* is the cornerstone of the OpenCL language. However, functional portability does not imply performance portability. That is, OpenCL allows developers to write kernels capable of running on all types of target hardware, but optimizing kernels for one type of target (e.g., GPU) does not guarantee the kernel will run efficiently on another target (e.g., CPU). With CPUs tending toward many cores, and the once special purpose, many-core GPUs offering general purpose functionality, it is easy to see that soon the CPU and GPU will meet somewhere in the middle as general purpose many-core architectures. Already, ATI has introduced the Fusion APU (Accelerated Processing Unit) which couples an AMD CPU and ATI GPU within a single die. OpenCL is an attempt to standardize programming ahead of this intersection.

Petascale computing centers around the world are leveraging GPU accelerators to achieve peak performance. In fact, many of today's high performance computing installations boast significantly more GPU accelerators than CPU counterparts. The Keeneland project is one such example, currently with 240 CPUs accompanied by 360 NVidia Fermi class GPUs with at least double that number expected by the end of 2012 [49].

Such throughput oriented architectures require developers to decompose problems into thousands of independent parallel tasks in order to fully harness the capabilities of the hardware. To this end, a plethora of research has been dedicated to researching algorithms in all fields of computational science. Of interest to us are methods for atmospheric- and geo-sciences.

3.1 Fragments

```
- what is new in the thesis (summarize: 1-2 pages)
- get all your figues
- 1/2 page description per figures
  All figure captions (self-contained). Don't skimp on words.
- Previous work
  GPUS/implicit methods
  GPUS/RBF
  RBF/Finite-Difference
  Preconditionners (info on that)
   Sparse matrix libraries
  Various GPU/based libraries
   Parallel implementations of RBF (there are any?)

Benchmarking....
   Timing (serial, parallel)
   Timing with special node reorderings. Explain logic
```

Test cases ...

Get all your references in bibtex. Send them to me.

Need detailed table of content.

As will be demonstrated in Chapter **??**, most of the literature surrounding the solution of PDEs with RBFs is based on collocation. Collocation finds an interpolant that passes through a set of *collocation points* (commonly chosen to coincide with RBF centers) satisfying the differential equations with zero residual. Collocation, then, is a global interpolation problem. Alternative methods exist based on local collocation formulations (see [**?**, **?**, **?**, **?**, **?**]). Recently, an new approach using FD-like stencils to approximate differential operators was proposed (see e.g., [**?**, **?**, **?**, **?**]). The so-called RBF-FD method uses RBFs to interpolate initial conditions, but not the differential equations—generalized FD stencils approximate differential operators.

Even today, RBFs are still up-and-coming in the scientific world with many avenues of research left to consider. Global formulations are understood to have spectral convergence properties, high accuracy and other benefits like adaptivity and ease of implementation over meshed methods [**?**]. However, little is known about the behavior of local and RBF-FD methods. Open questions include (but are in no way limited to): a) ideal node placement to eliminate singularities; b) data-structures for stencil storage and evaluation; c) problem sizes larger than a few thousand nodes; and d) parallel implementations across new heterogeneous multi- and many-core architectures. In response to this, our group, in collaboration with researchers assembled from a national lab and four universities (see Chapter **??**), has been granted funds by the National Science Foundation to collaboratively:

> *"Bring RBFs to the forefront of multi-scale geophysical modeling by developing fast, efficient, and parallelizable RBF algorithms in arbitrary geometries, with performance enhanced by hardware accelerators, such as graphic processing units (GPUs)."* [**?**]

In the last few years, GPUs transitioned from hardware dedicated to the embarrassingly parallel tasks involved in graphics rendering (e.g., rasterization) into multi-core coprocessors for high performance scientific computing. Thanks to the highly profitable and always demanding gaming industry, what began as a static rendering pipeline, was molded to allow fully dynamic execution with a SIMD-like programming model (Single Instruction Multiple Threads or SIMT). Changes in hardware were followed closely by evolving programming languages. Today, GPUs can be manipulated via high level languages similar to C/C++ and require no knowledge of computer graphics. In Chapters **??** and **??** we will discuss how GPU hardware and languages evolved to exceptionally higher compute capability than traditional CPUs, and became a popular platform for high performance computing.

True to our purpose statement above, the goal of this project is to integrate RBF methods for PDEs, Geophysics and large scale GPU computing. We begin in Chapter **??** with a discussion of related work on RBF-PDE methods, their applications, and related work on

GPUs. In Chapter **??** a formal introduction to RBFs for the solution of PDEs is provided. Chapter **??** considers various languages available for GPU computing and their appropriateness for our task. This is followed by a discussion in Chapter **??** of GPU hardware and the large scale clusters with integrated GPUs which will be used for heterogeneous multi-core computing. Finally, in Chapter **??** we present our proposal for research into efficient PDE solutions on multi-node, multi-GPU compute clusters using radial basis functions for Tsunami simulation.

3.2 RBF-FD as SpMV

From the definition of RBF-FD we can formulate the problem computationally in two ways.

Stencil operations are independent. Therefore in the first approach we can write kernels with perfect parallelism by writing kernels to dedicate a thread per stencil or a warp per stencil.

Unfortunately, perfect concurrency does not imply perfect or even ideal concurrency on the GPU.

We first demonstrate the case where one thread is dedicated to each stencil. This is followed by dedicating a group of thread to the stencil. In each case we are operating under the assumption that each stencil is independent on the GPU.

To further optimize RBF-FD on the GPU, we formulate the problem in terms of a Sparse Matrix-Vector Mulitply (SpMV). When we consider the problem in this light we generate a single Differentiation Matrix that can see two optimizations not possible with our stencil-based view:

- First, the sparse containers used in SpMV allow for their own unique optimizations to compress storage and leverage hardware cache.
- Evaluation of multiple derivatives can be accumulated by association into one matrix operation. This reduces the total number of floating point operations required per iteration.

CHAPTER 4

SPARSE MATRIX MULTIPLICATION ON THE GPU

With the generalization of RBF-FD derivative computation formulated as a sparse matrix multiplication, we can

CHAPTER 5

GMRES

The GMRES algorithm follows Algorithm **??**.

The Arnoldi process can be completed using in a variety of ways. Some libraries like CUSP and [**?**] prefer the straightforward Givens rotations because they are easily parallelizable. The givens algorithm is provided in Algorithm **??**

Others like [...] prefer to use an alternate algorithm. This algorithm is demonstrated in Algorithm **??**.

Parallelizing the algorithm is possible with communication points listed in red (*).

CHAPTER 6

VIENNACL

Author's Note: 4 pages

The ViennaCL project [41, 40] is a sparse matrix library built on OpenCL that provides templated C++ API to easily and efficiently solve large sparse matrix problems.

Our decision to utilize ViennaCL stems from its

The API provides containers for sparse and dense matrices, vectors and views. The sparse formats available are COO, CSR, (...) ELL and HYB

Matrix and Vector Views provide slices and subranges of containers. Subviews were recently added in version 1.2 with full functionality in v1.3. These views are essential to our work on multi-GPUs since we need to operate on views with full vectors containing the ghost values. (Show example of view).

Algorithms are provided for GMRES, CG, etc.

To build multi-GPU algorithms for GMRES we discard the ViennaCL arnoldi process in favor of the Givens rotations. Then we add MPI communication.

CHAPTER 7

ON THE FUTURE OF GPU COMPUTING

In 2006, NVidia released CUDA to the general public—a milestone representing the moment GPU computing reached the critical mass necessary for uptake into widespread and prolonged use. Since then attempts have been made to leverage the GPU for nearly ever field of scientific research.

The question, however, is frequently asked: is GPU computing a buzzword that will die out, or is it here to stay?

Fine tuning of kernels is the way of the past. Anyone fine tuning kernels will be operating at a lower level, attempting to optimize library routines underlying large scale codes. The fine tuning can ultimately be replaced by auto-tuned kernels (e.g., see work in ViennaCL to auto-tune similar to libATLAS).

Already this is seen in attempts to port applications to the GPU in so-called "minimally invasive" fashion.

So the short answer is: yes. And it is from this assumption that we proceed with our efforts to target the GPU with RBF-FD. However, we have made a few predictions on the future of GPUs and these predictions guided our efforts to port RBFs onto GPUs.

First, we are proponents for OpenCL over CUDA. Although CUDA has a large following due to its early release, we believe in functional portability of code enabled by OpenCL over the performance provided by CUDA.

7.1 OpenCL vs CUDA

We support opencl. The difference lying in the end-goal of supporting a wide range of hardware with a compiler as support for porting.

7.2 Pragmas

7.2.1 PGI Accelerator Pragmas

During a 2009 summer internship at NCAR, we investigated the use of Portland Group Inc (PGI) Accelerator pragmas to accelerate a subgrid physics module for resolving clouds in the Community Climate Simulation Models (CCSM). The results were sketchy(*) for numerous reasons. First, as early adopters, we were attempting to

7.2.2 OpenACC

7.3 MPI and GPU Computing

CUDA 5 introduced support for MPI directly from the GPU. But we are proponents for OpenCL, not CUDA. The difference lying in the

CHAPTER 8

COMMUNITY OUTREACH

-
-
- E. F. Bollig. Multi-GPU Solutions of Hyperbolic and Elliptic PDEs with RBF-FD. Contributed Talk. SIAMSEAS Annual Meeting. Huntsville, AL, March 2012
- E. F. Bollig. Parallel Algorithms for RBF-FD Solutions to PDEs on the Sphere. Poster. FSU Dept. of Sci. Comp. Computational Xpo. Tallahassee, FL, March 2012
- E. F. Bollig. A Multi-CPU/GPU implementation of RBF-generated finite differences for PDEs on a Sphere. Poster. American Geophys. Union. San Francisco, CA, December 2011
- E. F. Bollig. Accelerating RBF-FD in Parallel Environments. Student Presentation. NSF CBMS Workshop: Radial Basis Functions Mathematical Developments and Applications. Darmouth, MA, June 2011

TODO: add survey of goals stated in prospectus and whether they were completed or not (Probably part of slides, not actual dissertation)

BIBLIOGRAPHY

[1] HMPP Data Sheet. http://www.caps-entreprise.com/upload/article/fichier/64fichier1.pdf, 2009. 8

[2] AccelerEyes. *Jacket User Guide - The GPU Engine for MATLAB*, 1.2.1 edition, November 2009. 7

[3] Sylvie Barak. Gpu technology key to exascale says nvidia. http://www.eetimes.com/electronics-news/4230659/GPU-technology-key-to-exascale-says-Nvidia, November 2011. 1

[4] E. F. Bollig. A Multi-CPU/GPU implementation of RBF-generated finite differences for PDEs on a Sphere. Poster. American Geophys. Union. San Francisco, CA, December 2011.

[5] E. F. Bollig. Accelerating RBF-FD in Parallel Environments. Student Presentation. NSF CBMS Workshop: Radial Basis Functions Mathematical Developments and Applications. Darmouth, MA, June 2011.

[6] E. F. Bollig. Multi-GPU Solutions of Hyperbolic and Elliptic PDEs with RBF-FD. Contributed Talk. SIAMSEAS Annual Meeting. Huntsville, AL, March 2012.

[7] E. F. Bollig. Parallel Algorithms for RBF-FD Solutions to PDEs on the Sphere. Poster. FSU Dept. of Sci. Comp. Computational Xpo. Tallahassee, FL, March 2012.

[8] Evan F. Bollig, Natasha Flyer, and Gordon Erlebacher. Solution to pdes using radial basis function finite-differences (rbf-fd) on multiple gpus. *Journal of Computational Physics*, (0):–, 2012. 6

[9] Andreas Brandstetter and Alessandro Artusi. Radial Basis Function Networks GPU Based Implementation. *IEEE Transaction on Neural Network*, 19(12):2150–2161, December 2008. 7

[10] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell. Smooth Surface Reconstruction from Noisy Range Data. In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 119–ff, New York, NY, USA, 2003. ACM. 7

[11] Tom Cecil, Jianliang Qian, and Stanley Osher. Numerical Methods for High Dimensional Hamilton-Jacobi Equations Using Radial Basis Functions. *JOURNAL OF COMPUTATIONAL PHYSICS*, 196:327–347, 2004. 5, 6

[12] G Chandhini and Y Sanyasiraju. Local RBF-FD Solutions for Steady Convection-Diffusion Problems. *International Journal for Numerical Methods in Engineering*, 72(3), 2007. 4, 6

[13] P P Chinchapatnam, K Djidjeli, P B Nair, and M Tan. A compact RBF-FD based meshless method for the incompressible Navier–Stokes equations. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 223(3):275–290, March 2009. 6

[14] A Corrigan and HQ Dinh. Computing and Rendering Implicit Surfaces Composed of Radial Basis Functions on the GPU. *International Workshop on Volume Graphics*, 2005. 7

[15] N Cuntz, M Leidl, TU Darmstadt, GA Kolb, CR Salama, M Böttinger, D Klimarechenzentrum, and G Hamburg. GPU-based Dynamic Flow Visualization for Climate Research Applications. *Proc. SimVis*, pages 371–384, 2007. 7

[16] E Divo and AJ Kassab. An Efficient Localized Radial Basis Function Meshless Method for Fluid Flow and Conjugate Heat Transfer. *Journal of Heat Transfer*, 129:124, 2007. 7

[17] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Rev.*, 41(4):637–676, 1999. 3

[18] Gregory E. Fasshauer. *Meshfree Approximation Methods with MATLAB*, volume 6 of *Interdisciplinary Mathematical Sciences*. World Scientific Publishing Co. Pte. Ltd., 5 Toh Tuck Link, Singapore 596224, 2007. 3, 4

[19] Natasha Flyer and Bengt Fornberg. Radial basis functions: Developments and applications to planetary scale flows. *Computers & Fluids*, 46(1):23–32, July 2011. 4

[20] Natasha Flyer and Erik Lehto. Rotational transport on a sphere: Local node refinement with radial basis functions. *Journal of Computational Physics*, 229(6):1954–1969, March 2010. 4

[21] Natasha Flyer, Erik Lehto, Sebastien Blaise, Grady B. Wright, and Amik St-Cyr. Rbf-generated finite differences for nonlinear transport on a sphere: shallow water simulations. *Submitted to Elsevier*, pages 1–29, 2011. 6

[22] Natasha Flyer and Grady B. Wright. Transport schemes on a sphere using radial basis functions. *Journal of Computational Physics*, 226(1):1059 – 1084, 2007. 4

[23] Natasha Flyer and Grady B. Wright. A Radial Basis Function Method for the Shallow Water Equations on a Sphere. In *Proc. R. Soc. A*, volume 465, pages 1949–1976, December 2009. 4, 5

[24] B. Fornberg and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Computers & Mathematics with Applications*, 48(5-6):853 – 867, 2004. 5

[25] Bengt Fornberg and Natasha Flyer. Accuracy of Radial Basis Function Interpolation and Derivative Approximations on 1-D Infinite Grids. *Adv. Comput. Math*, 23:5–20, 2005. 5

[26] Bengt Fornberg, Elisabeth Larsson, and Natasha Flyer. Stable Computations with Gaussian Radial Basis Functions. *SIAM J. on Scientific Computing*, 33(2):869—-892, 2011. 5

[27] Bengt Fornberg and Erik Lehto. Stabilization of RBF-generated finite difference methods for convective PDEs. *Journal of Computational Physics*, 230(6):2270–2285, March 2011. 6

[28] Bengt Fornberg and Cécile Piret. A Stable Algorithm for Flat Radial Basis Functions on a Sphere. *SIAM Journal on Scientific Computing*, 30(1):60–80, 2007. 5

[29] Bengt Fornberg and Cécile Piret. On Choosing a Radial Basis Function and a Shape Parameter when Solving a Convective PDE on a Sphere. *Journal of Computational Physics*, 227(5):2758 – 2780, 2008. 5

[30] E. J. Fuselier and G. B. Wright. A High-Order Kernel Method for Diffusion and Reaction-Diffusion Equations on Surfaces. *ArXiv e-prints*, May 2012. 3

[31] R Hardy. Multiquadratic Equations of Topography and Other Irregular Surfaces. *J. Geophysical Research*, (76):1–905, 1971. 4

[32] A. Iske. *Multiresolution Methods in Scattered Data Modeling*. Springer, 2004. 3, 4

[33] E J Kansa. Multiquadrics–A scattered data approximation scheme with applications to computational fluid-dynamics. I. Surface approximations and partial derivative estimates. *Computers Math. Applic*, (19):127–145, 1990. 4, 8

[34] Khronos OpenCL Working Group. *The OpenCL Specification (Version: 1.0.48)*, October 2009. 8

[35] G Kosec and B Šarler. Solution of thermo-fluid problems by collocation with local pressure correction. *International Journal of Numerical Methods for Heat & Fluid Flow*, 18, 2008. 7

[36] Elisabeth Larsson and Bengt Fornberg. A Numerical Study of some Radial Basis Function based Solution Methods for Elliptic PDEs. *Comput. Math. Appl*, 46:891–902, 2003. 5, 8

[37] Shaofan Li and Wing K. Liu. *Meshfree Particle Methods*. Springer Publishing Company, Incorporated, 2007. 3, 4

[38] NVidia. *NVIDIA CUDA - NVIDIA CUDA C - Programming Guide version 4.0*, March 2011. 8

[39] Portland Group Inc. *CUDA Fortran Programming Guide and Reference*, 1.0 edition, November 2009. 8

[40] Karl Rupp, Florian Rudolf, and Josef Weinbub. ViennaCL-A High Level Linear Algebra Library for GPUs and Multi-Core CPUs. In *Proc. GPUScA*, pages 51—56, 2010. 14

[41] Karl Rupp, Josef Weinbub, and Florian Rudolf. Automatic Performance Optimization in ViennaCL for GPUs Categories and Subject Descriptors. In *Proceedings of the 9th Workshop on Parallel/High-Performance Object-Oriented Scientific Computing*, pages 6:1–6:6. ACM, 2010. 14

[42] Robert Schaback. Multivariate Interpolation and Approximation by Translates of a Basis Function. In C.K. Chui and L.L. Schumaker, editors, *Approximaton Theory VIII–Vol. 1: Approximation and Interpolation*, pages 491–514. World Scientific Publishing Co., Inc, 1995. 5

[43] J. Schmidt, C. Piret, B.J. Kadlec, D.A. Yuen, E. Sevre, N. Zhang, and Y. Liu. Simulating Tsunami Shallow-Water Equations with Graphics Accelerated Hardware (GPU) and Radial Basis Functions (RBF). In *South China Sea Tsunami Workshop*, 2008. 6

[44] J. Schmidt, C. Piret, N. Zhang, B.J. Kadlec, D.A. Yuen, Y. Liu, G.B. Wright, and E. Sevre. Modeling of Tsunami Waves and Atmospheric Swirling Flows with Graphics Processing Unit (GPU) and Radial Basis Functions (RBF). *Concurrency and Computat.: Pract. Exper.*, 2009. 6, 7

[45] C. Shu, H. Ding, and K. S. Yeo. Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 192(7-8):941 – 954, 2003. 5, 6

[46] D Stevens, H Power, M Lees, and H Morvan. The use of PDE centres in the local RBF Hermitian method for 3D convective-diffusion problems. *Journal of Computational Physics*, 2009. 6

[47] A. I. Tolstykh and D. A. Shirobokov. On using radial basis functions in a "finite difference mode" with applications to elasticity problems. In *Computational Mechanics*, volume 33, pages 68 – 79. Springer, December 2003. 5

[48] A.I. Tolstykh. On using RBF-based differencing formulas for unstructured and mixed structured-unstructured grid calculations. In *Proceedings of the 16 IMACS World Congress, Lausanne*, pages 1–6, 2000. 5

[49] J.S. Vetter, R. Glassbrook, J. Dongarra, K. Schwan, B. Loftis, S. McNally, J. Meredith, J. Rogers, P. Roth, K. Spafford, and S. Yalamanchili. Keeneland: Bringing heterogeneous GPU computing to the computational science community. *IEEE Computing in Science and Engineering*, 13(5):90–95, 2011. 9

[50] M Weiler, R Botchen, S Stegmaier, T Ertl, J Huang, Y Jang, DS Ebert, and KP Gaither. Hardware-Assisted Feature Analysis and Visualization of Procedurally Encoded Multifield Volumetric Data. *IEEE Computer Graphics and Applications*, 25(5):72–81, 2005. 7

[51] Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4:389–396, 1995. 10.1007/BF02123482. 5

[52] Grady B. Wright. *Radial Basis Function Interpolation: Numerical and Analytical Developments*. PhD thesis, University of Colorado, 2003. 5

[53] Grady B. Wright, Natasha Flyer, and David A. Yuen. A hybrid radial basis function–pseudospectral method for thermal convection in a 3-d spherical shell. *Geochem. Geophys. Geosyst.*, 11(Q07003):18 pp., 2010. 4

[54] Rio Yokota, L.A. Barba, and Matthew G. Knepley. PetRBF — A parallel O(N) algorithm for radial basis function interpolation with Gaussians. *Computer Methods in Applied Mechanics and Engineering*, 199(25-28):1793–1804, May 2010. 7