PERSONAL INFORMATION

Applicant ID: 1000245690
Prefix:
First Name: Louis
Middle Name: Peter
Last Name: Jenkins
Suffix: Jr
Previous Last Name 1:
Previous Last Name 2:
ORCID Identifier:

Mailing Address

Street Address: 60 Crittenden Blvd
Apt 110
City: Rochester
State: NY
Zip Code: 14620
Country: United States
E-mail: LouisJenkinsCS@hotmail.com
Phone Number: 6109311207

Permanent Address

Same as mailing address: Y

Date of Birth

Date of Birth: 12/11/1993
State: PA
Country: United States
Citizenship: US Citizen
If permanent resident alien, date status was granted:

High School Location

City: Drexel Hill
State: PA
Country: United States

Demographic Information

Gender: Male
Veteran Status: No
Ethnicity: Not Hispanic or Not Latino
Race: Black or African-American
Disability: No

EDUCATION AND WORK EXPERIENCE

List colleges or universities attended and your enrollment details.

College/Univ.	Location	Start Date	End Date	Degree Granting Program	Degree	Degree Cmpl.	Grad. Date	Field of Study	Cum. GPA	GPA Basis
Bloomsburg University of Pennsylvania	BLOOMSBURG, PA, United States	05/2012	12/2017	Yes	BS	Yes	12/2017	Comp/IS/Eng - Computational Science and Engineering	3.06	4.0
University of Rochester	ROCHESTER, NY, United States	08/2018	04/2023	Yes	PhD	No, still enrolled in program		Comp/IS/Eng - Computational Science and Engineering	4.0	4.0

List teaching and work experiences relevant to your field of study since entering college/university. Experiences do not have to be limited to the academic realm.

Title	Institution/Organization	Start Date	Other Experience Ongoing	End Date
Developer	Google Summer of Code - Chapel, Cray Inc	04/2017	No	09/2017
Research Assistant	Lehigh University	04/2016	No	07/2016
Tech. Intern, Level 4	Pacific Northwest National Laboratory	04/2018	No	07/2018
PhD Intern, External Collaborator	Pacific Northwest National Laboratory	07/2018	Yes	

List any significant academic honors, fellowships, scholarships, publications and presentations.

**[2018] Tech. Intern, Level 4 **

Organization: Pacific Northwest National Laboratory

Mentor: Andrew Lumsdaine

Project: Chapel HyperGraph Library (CHGL)

Publication #1 [Authors]: L. Jenkins, M. Zalewski, S. Aksoy, H. Medal, C. Joslyn, et al.

Publication #1 [Title]: "Chapel HyperGraph Library (CHGL)"

Publication #1 [Conference]: 2018 IEEE High Performance Extreme Computing Conference (HPEC-2018), Twenty-second Annual HPEC Conference

Summary: Developed the first, to my knowledge, exascale hypergraph library. Joint collaborative effort between Mississippi State University and Pacific Northwest National Laboratory, where I was the sole developer of the project. Project intended to explore Chapel's ability to handle irregular applications and to address shortcomings and bugs in the language. Government sponsor.

Abstract: (See Below)

"We present the Chapel Hypergraph Library (CHGL), a library for hypergraph computation in the emerging Chapel language. Hypergraphs generalize graphs, where a hypergraph edge can connect any number of vertices. Thus, hypergraphs capture high-order, high-dimensional interactions between multiple entities that are not directly expressible in graphs. CHGL is designed to provide HPC-class computation with high-level abstractions and modern language support for parallel computing on shared memory and distributed memory systems. In this paper we describe the design of CHGL, including ﬁrst principles, data structures, and algorithms, and we present preliminary performance results based on a graph generation use case. We also discuss ongoing work of co-design with Chapel, which is currently centered on improving performance."

Publication #2 [Authors]: L. Jenkins, M. Zalewski, M. Ferguson

Publication #2 [Title]: Chapel Aggregation Library (CAL) ~To Appear~

Publication #2 [Conference]: Parallel Applications Workshop, Alternatives To MPI (SC'18 Workshop)

Summary: Developed an aggregation library that addressed issue of fine-grained communications inherent in irregular applications. Library is written entirely in Chapel, and was a collaborative project between PNNL and Cray. Design is based on the AM++ library and makes use of Chapel language features to make it easy to use.

Abstract: (See Below)

"Fine-grained communication is a fundamental principle of the Partitioned Global Address Space (PGAS), which serves to simplify creating and reasoning about programs in the distributed context. However, per-message overheads of communication rapidly accumulate in programs that generate a high volume of small messages, limiting the effective bandwidth and potentially increasing latency if the messages are generated at a much higher rate than the effective network bandwidth. One way to reduce such ﬁne-grained communication is by coarsening the granularity by aggregating data, or by buffering the smaller communications together in a way that they can be handled in bulk. Once these communications are buffered, the multiple units of the aggregated data can be combined into fewer units in an optimization called coalescing.

The Chapel Aggregation Library (CAL) provides a straightforward approach to handling both aggregation and coalescing of data in Chapel and aims to be as generic and minimal as possible to maximize code reuse and minimize its increase in complexity on user applications. CAL provides a high-performance, distributed, and parallel-safe solution that is entirely written as a Chapel module. In addition to being easy to use, CAL improves the performance of some benchmarks by one to two orders of magnitude over naive implementations at 32 compute-nodes on a Cray XC50."

[2018] Independent Researcher

Publication [Authors]: L. Jenkins

Publication [Title]: RCUArray: An RCU-like Parallel-Safe Distributed Resizable Array

Publication [Conference]: The 5th Annual Chapel Implementers and Users Workshop (CHIUW 2018, IPDPSW)

Summary: Shortly after becoming acquainted with the idea of RCU, I designed a parallel-safe array, distributed over multiple compute nodes, that can be read and written to while being resized. Intended to be used as the base for a descriptor table or other distributed data structure such as a distributed vector.

Abstract (See Below):

"Presented in this work is RCUArray, a parallel-safe distributed array that allows for read and update operations to occur concurrently with a resize via Read-Copy-Update. Also presented is a novel extension to Epoch-Based Reclamation (EBR) that functions without the requirement for either Task-Local or Thread-Local storage, as the Chapel language currently lacks a notion of either. Also presented is an extension to Quiescent State-Based Reclamation (QSBR) that is implemented in Chapel?s runtime and allows for parallel-safe memory reclamation of arbitrary data. At 32-nodes with 44-cores per node, the RCUArray with EBR provides only 20% of the performance of an unsynchronized Chapel block distributed array for read and update operations but near-equivalent with QSBR; in both cases RCUArray is up to 40x faster for resize operations."

[2017] Google Summer of Code

Mentor: Engin Kayraklioglu, Michael Ferguson

Organization: Chapel (Cray Inc)

Project: Distributed Data Structures

Summary: First time exposure to parallel-distributed programming language Chapel and the HPC abstraction PGAS, created two of the first novel data structures that scale to at least 2 orders of magnitude for a moderate cluster (~3072 Processors).

Abstract: (See Below)

"Built-in data structures are a necessity for any budding language, and in a language where distributed computing is at its core, data structures that can properly be maintained across clusters is desired.

For my project, I have designed the core framework for a distributed data structures library, and have implemented two novel scalable distributed data structures, an ordered deque and an unordered multiset, that exceed a naive implementation by at least two orders of magnitude in a moderately sized cluster."

[2016] Research Assistant

Mentor: Michael Spear

Organization: Lehigh University

Project: Concurrent and Scalable Built-in Hash-Table for the Go Programming Language

Award #1: Peer's Choice for Outstanding Project

Award #2: Honorable Mention for 2017 Computing Research Association Outstanding Undergraduate Researchers

Publication [Authors]: L. Jenkins, T. Zhou, M. Spear

Publication [Title]: "Redesigning Go's Built-In Map to Support Concurrent Operations"

Publication [Conference]: The 26th International Conference on Parallel Architectures and Compilation Techniques (PACT 2017)

Summary: First time exposure to Go programming language, created a built-in concurrent-safe hash table that was incrementally designed, became a high-performance scalable and novel map that provided more features and stronger guarantees than other state-of-the-art maps in research literature.

Abstract: (See Below)

"The Go language lacks built-in data structures that allow fine-grained concurrent access. In particular, its map data type, one of only two generic collections in Go, limits concurrency to the case where all operations are read-only; any mutation (insert, update, or remove) requires exclusive access to the entire map. The tight integration of this map into the Go language and runtime precludes its replacement with known scalable map implementations.

This paper introduces the Interlocked Hash Table (IHT). The IHT is the result of language-driven data structure design: it requires minimal changes to the Go map API, supports the full range of operations available on the sequential Go map, and provides a path for the language to evolve to become more amenable to scalable computation over shared data structures. The IHT employs a novel optimistic locking protocol to avoid the risk of deadlock, and allows large critical sections that access a single IHT element, and can easily support multikey atomic operations. These features come at the cost of relaxed, though still straightforward, iteration semantics. In experimentation in both Java and Go, the IHT performs well, reaching up to 7× the performance of the state of the art in Go at 24 threads. In Java, the IHT performs on par with the best Java maps in the research literature, while providing iteration and other features absent from other maps."

Baccalaureate Institution: Bloomsburg University of Pennsylvania
Current Institution: University of Rochester
Are you or have you been in a joint baccalaureate-master's degree program? No

PROPOSED FIELD OF STUDY

Field of Study: Comp/IS/Eng - Computational Science and Engineering

Is your proposed graduate study interdisciplinary? No

PROPOSED GRADUATE STUDY

Proposed University or College: University of Rochester
Proposed Graduate Program: PhD
City: Rochester
State: NY
Country: United States

REFERENCES

List names and organizational affiliations of individuals submitting Letters of Reference (at least three reference letters must be received by the published deadline for the application to be complete).

Last Name	First Name	MI	Organization	E-mail Address	Ref. Rank	Status
Lumsdaine	Andrew		Pacific Northwest National Laboratory	andrew.lumsdaine@pnnl.gov	5	Unsubmitted
Zalewski	Marcin		Pacific Northwest National Laboratory	marcin.zalewski@pnnl.gov	2	Received
Spear	Michael		Lehigh University	spear@cse.lehigh.edu	4	Unsubmitted
Ferguson	Michael		Cray Inc.	mferguson@cray.com	1	Received
Scott	Michael	L	University of Rochester	scott@cs.rochester.edu	3	Received

PERSONAL, RELEVANT BACKGROUND AND FUTURE GOALS STATEMENT

Please outline your educational and professional development plans and career goals. How do you envision graduate school preparing you for a career that allows you to contribute to expanding scientific understanding as well as broadly benefit society?

Describe your personal, educational and/or professional experiences that motivate your decision to pursue advanced study in science, technology, engineering or mathematics (STEM). Include specific examples of any research and/or professional activities in which you have participated. Present a concise description of the activities, highlight the results and discuss how these activities have prepared you to seek a graduate degree. Specify your role in the activity including the extent to which you worked independently and/or as part of a team. Describe the contributions of your activity to advancing knowledge in STEM fields as well as the potential for broader societal impacts (See Solicitation, Section VI, for more information about Broader Impacts).

NSF Fellows are expected to become globally engaged knowledge experts and leaders who can contribute significantly to research, education, and innovations in science and engineering. The purpose of this essay is to demonstrate your potential to satisfy this requirement. Your ideas and examples do not have to be confined necessarily to the discipline that you have chosen to pursue.

If you have completed more than one academic year in a graduate degree-granting program or a graduate or professional degree, followed by an interruption of at least two consecutive years, please address the reasons for the interruption in graduate study here.

Document Uploaded: Yes

GRADUATE RESEARCH PLAN STATEMENT

Present an original research topic that you would like to pursue in graduate school. Describe the research idea, your general approach, as well as any unique resources that may be needed for accomplishing the research goal (i.e., access to national facilities or collections, collaborations, overseas work, etc.) You may choose to include important literature citations. Address the potential of the research to advance knowledge and understanding within science as well as the potential for broader impacts on society. The research discussed must be in a field listed in the Solicitation (Section X, Fields of Study).

Document Uploaded: Yes

Proposed Research Title

The title should be brief, informative, scientifically or technically valid, intelligible to a scientifically or technically literate reader, and suitable for use in the public press. It should describe in succinct terms your proposed research, reflecting the contents of your proposal. Use key words, and do not use abbreviations and chemical formulas(in 255 characters or less). This title will be used for searching research topics using the key words you supply. Do not use curly brackets, {}, in your Proposed Research Title or Key Words.

Proposed Research Title: Scalable Software Atomics for the Partitioned Global Address Space

Use key words to describe the Graduate Research Plan Statement (in 50 characters or less).

Key Words: Distributed, Partitioned Global Address Space

NSF GRFP PROGRAM INFORMATION

Select the level that most appropriately describes your stage of study at the GRFP application deadline. All enrollment in graduate or professional degree-granting programs must be included.

First-year graduate student currently enrolled in a graduate degree-granting program, who has never applied to GRFP before as a graduate student or returning graduate student.

Advisor

If you are currently enrolled in graduate school (levels 2 or 3), provide the name(s) of your current or potential graduate research advisor(s). If you do not have a current or potential graduate research advisor, provide the contact information of your graduate program director. Entry of at least one advisor is required with a maximum of three.

First Name	MI	Last Name	E-mail Address
Michael		Scott	scott@cs.rochester.edu

NSF publishes the names, the baccalaureate and current institutions, and the fields of study of Fellowship recipients and Honorable Mention List on FastLane.

Do you wish your name to be published on the Honorable Mention List, posted at https://www.research.gov/grfp/?: Yes

Louis Jenkins

Personal Statement

Introduction

Growing up, technology was a luxury and privilege affordable to very few single-mother households, and it wasn't until the age of 12 that I obtained my first personal computer. As I lacked a role model, something as foreign as computer programming felt like an 'elite' skill that I just wasn't smart enough to develop on my own. It wasn't until the age of 18 that I decided to confront my fear by gaining exposure to computer programming, and then to overcome my fear by committing and declaring my major in Computer Science at the age of 21, with aspirations to become a Software Engineer. I did not begin my journey as a Computer Scientist until the age of 22, when I obtained my first research experience, award, and publication. Now, at the age of 24, I am the lead author on 4 publications, had a rich set of experiences in academic and industry-related research, and am a part-time government contractor for the Department of Energy.

Experiences

In the Summer of 2016, I was fortunate enough to partake in a 10-week National Science Foundation sponsored Research Experience for Undergraduates (REU) at Lehigh University. It was at this time I was introduced to Prof. Michael F. Spear, who became the supervisor on my research project: "Concurrent and Scalable Built-In Hash Table for the Go Programming Language." The project centered around replacing Go's sequential map with a concurrent one, which involved making significant changes to Go's runtime and compiler. When Prof. Spear stated that the project "probably wasn't even possible," it filled me with a sense of passion and purpose for the very first time in my life; I would not only do something novel but also of great importance! Despite never having used the Go programming language prior to that point, despite having no experience in compilers or runtime systems, and despite having only self-taught experience in the area of concurrency, by the end of the REU, I had completed a functional and scalable prototype. Despite working alone when my peers worked in groups of 2 or 3, my project was awarded "Peer's Choice for Outstanding Project." My research with Prof. Spear did not end there: after the REU was over, we further improved the design to the point that it was published with me as lead author under the title "Redesigning Go's Built-In Map to Support Concurrent Operations" at the *26th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. Later, thanks to a push from Prof. Spear, I was nominated by my home institution, Bloomsburg University, and received Honorable Mention in the Computing Research Association's Outstanding Undergraduate Researchers competition for 2017, which was sponsored by Microsoft Research that year.

During the Summer of 2017, I was accepted into the Google Summer of Code program to work on the Chapel programming language. I was introduced to my two mentors, Engin Kayraklioglu, a previous intern at Cray Inc., and Michael Ferguson, a software engineer at Cray, where I worked remotely on my project: "Distributed Data Structures." The project centered around implementing distributed queues in Chapel that would scale in the Partitioned Global Address Space (PGAS), though I had no prior experience in either. The project was open-ended enough to allow me to explore as much as I wanted and was made all the more fun when I was provided access to a shared Cray XC40 supercomputer on which to perform my experiments. Despite taking 3 courses concurrently with the project, by the end of the program I had

successfully built not only an interface for creating more distributed data structures, but also a decentralized segmented queue with work stealing and a more novel centralized segmented ordered queue that makes use of Remote Direct Memory Access (RDMA) atomics. Both queues scale linearly with both the number of compute nodes and the number of processors per node. The interface and both data structures are bundled as package modules with Chapel as of version 1.16.

During the Fall of 2017, I took part in an undergraduate research project at Bloomsburg University which was awarded a modest research grant provided by the school. The project, inspired by Azul Systems' Zing, was to create an open source Java Virtual Machine (JVM) written in Haskell that performed Just-in-Time compilation of JVM bytecode to the intermediate representation of the LLVM compiler infrastructure (LLVM IR). The project taught me the process of converting a stack-based machine to a register-based machine. The prototype was capable of mapping simple programs written in Java down to LLVM IR with full access to LLVM's many optimizations. To demonstrate this to an audience, I devised a Graphical User Interface capable of showing the conversion of Java down to bytecode, from bytecode to unoptimized LLVM IR, and from unoptimized LLVM IR to various levels of optimized LLVM IR.

In the Spring of 2018, after graduating from Bloomsburg University with my B.S. degree, I continued the work I had done during my Google Summer of Code. During that time, I had experimented with the Read-Copy-Update synchronization technique and its potential applications in the Partitioned Global Address Space. While doing so, I had explored Quiescent State-Based Reclamation and Epoch-Based Reclamation, two concurrent-safe memory reclamation techniques. The former I extended to make use of epochs and implemented it in Chapel's runtime, and the latter to work without the need for thread-local storage so that it could work in Chapel. Combining both memory management techniques and with Read-Copy-Update, I designed a novel distributed array that was safe to read and write in parallel while being resized. With the encouragement from Brad Chamberlain, a Principal Engineer at Cray and lead designer of the Chapel programming language, the scalable design and all three mechanisms were featured in a solo-authored paper titled "RCUArray: An RCU-like Parallel-Safe Distributed Resizable Array" in the *5th Annual Chapel Implementers and Users Workshop (CHIUW 2018).*

In the Summer of 2018, I was endorsed by Brad Chamberlain and recommended by Prof. Spear to Andrew Lumsdaine, co-founder of MPI and Chief Scientist at the Northwest Institute for Advanced Computing, to work on a government-sponsored collaborative project between the Department of Energy's Pacific Northwest National Laboratory (PNNL) and Mississippi State University. The project involved designing and implementing an exascale hypergraph library written in the Chapel programming language – a project for which I became the lead and later sole developer. The project, titled the "Chapel HyperGraph Library (CHGL)" was featured in a self-titled publication with me as lead author in the *2018 IEEE High Performance Extreme Computing Conference (HPEC '18).* To overcome the issue of excessive fine-grained communication, I have collaborated with my former mentor, Michael Ferguson, to devise the "Chapel Aggregation Library (CAL)," which is also to be featured in a self-titled work, also with me as lead author, in the *Parallel Applications Workshop, Alternatives to MPI (PAW-ATM).* During the summer and into the Fall, I also volunteered as a coach for a Rail's Girls Summer of Code team alongside my former mentors, Engin Kayraklioglu and Michael Ferguson. I helped

guide two undergraduates on their project, which centered around designing and implementing a distributed sorting library for Chapel.

As of Fall 2018, I am a PhD student at the University of Rochester; working with Prof. Michael Scott, who was Prof. Michael Spear's own PhD advisor. As a Research Assistant, I am working with Prof. Scott to develop a testing tool that aims to verify correctness for crash-resilient data structures in nonvolatile memory. I will also be working part-time as a government contractor, continuing my work as lead developer of the "Chapel HyperGraph Library".

Conclusion

For much of my life I have been afraid to try new things and to commit to them. As a result, I have missed out on many opportunities. Now it is my desire to turn over a new leaf, to bloom and to blossom, and to do as my mentor Prof. Michael Spear had said to: "shoot for the moon to land among the stars". I strive to become a great computer scientist and I am doing everything I can to seize the opportunities that come my way. I believe that I have the potential to become a leader in my scientific discipline, and I would be grateful for NSF's assistance.

Scalable Software Atomics for the Partitioned Global Address Space

The *Partitioned Global Address Space* (PGAS), a memory model in which the global address space is explicitly partitioned across compute nodes in a cluster, strives to bridge the gap between shared-memory and distributed-memory programming. Accesses to *remote* memory allocated on another compute node are performed through one-sided *PUT* and *GET* operations, which are forms of *Remote Direct Memory Access* (RDMA), handled by specialized hardware instead of the CPU. In the Chapel [1,2] programming language, loads and stores that are not proven to be *local* at compile time are transformed into PUT and GET operations, enabling shared-memory applications to function in distributed-memory environments, and vice-versa. While these transformations simplify programming, they do not come without cost, as pointers to potentially remote memory must be *widened* to 128 bits to contain the virtual-address as well as the unique identifier, or *rank*, of the compute node on which the memory is allocated. One immediate drawback is the added difficulty this poses for portable atomic operations on widened pointers. While many modern processors support 128-bit atomic compare-exchange operations, RDMA atomic operations currently only support up to 64-bit atomic operations. RDMA atomics prove to be over an order of magnitude faster than processor atomics performed over *Remote Procedure Calls* (RPC).

A prototype to a software solution that I have designed [3,4] makes use of a distributed and parallel-safe resizable array by making use of *Read-Copy-Update* (RCU) [5], which is a synchronization technique that prioritizes performance of readers over the performance of writers. This array, titled *RCUArray*, serves as a table for *descriptors* that can be used in place of widened pointers. A descriptor, much like an operating systems file descriptor, simply describes some arbitrary data by acting as an index into the table. As a widened pointer is too large to be used in atomic operations, a 64-bit descriptor can be used in place of the pointer. A pointer must first be registered to obtain a descriptor, where that descriptor is always local to the compute node and can be used to locate the pointer from the table. RCUArray allows the table to be resized on-demand if no descriptor can be *recycled* during registration, as shown in Figure 1, and in a way that will not impede the progress of other concurrent operations, hence maintaining scalability. A descriptor is recycled when it is unregistered from the table after it is no longer in use and is placed in the recycle list of the compute node for which it was allocated. Recycling of descriptors can be seen as the equivalent of memory deallocation in that it is not safe to access a descriptor once it has been recycled.

Following from Chapel's *multiresolution design philosophy* [6], where higher-level abstractions are written in terms of lower-level abstractions and the users are free to use either at their discretion, atomics of widened pointers can serve as *building blocks* for higher-level abstractions. One abstraction that can be improved is safe memory reclamation, which becomes necessary when memory can be accessed at any time from any compute node. I have conducted previous work to integrate low-overhead memory reclamation algorithms into Chapel [5,7], but further improvement can be achieved in future works using atomics. By utilizing both atomics and safe memory reclamation, we can create scalable distributed data structures as the next level of abstraction. In previous work, I have implemented two distributed data structures [8]: a decentralized, segmented, work-stealing queue and a centralized RDMA Fetch-And-Add atomic based, segmented, ordered queue. While both queues scale well, their designs can be overhauled and improved with the availability of atomics and safe memory reclamation to achieve greater

performance. These building blocks can be used to build higher-level and more user-friendly abstractions and data structures.

In the long term, a hardware solution can later be devised for 128-bit RDMA atomics, but in the meantime, a software solution is not only capable of providing insight into the potential gain from solving this problem, but also can serve as a fallback for the sake of portability. The scalable higher-level and user-friendly abstractions that may result from a solution to the problem of performing atomics on widened pointers can be used to enable a more object-oriented approach to High-Performance Computing (HPC) applications. This object-oriented approach is not only general and domain-independent, but more familiar to programmers coming from other languages such as Java, C#, and Python. As consequence, this lowers the barrier for entry into HPC, shortens the development time of scientific computing applications, and can potentially speed up future scientific discovery!

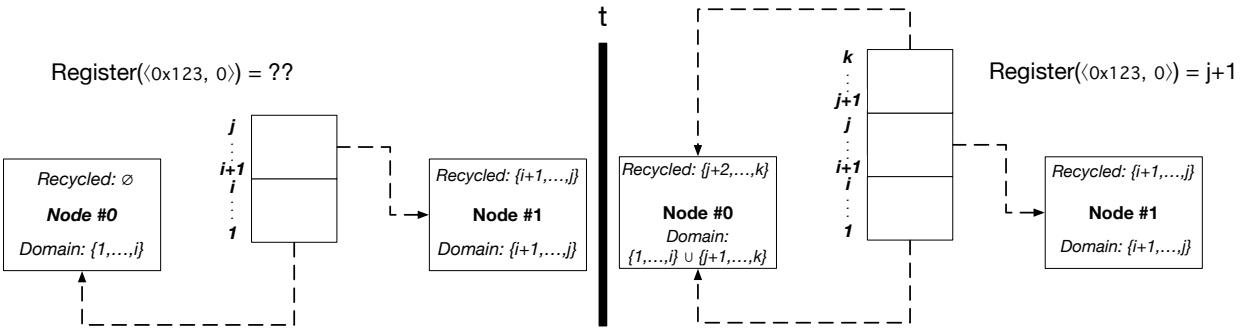

Figure 1. In this example, a widened pointer with virtual address 0x123 and rank 0 will be registered in the table. Recycled represents a set of recycled descriptors, Domain represents the portion of the distributed array that is allocated on that node, and the dashed arrows points to the Node that the block is distributed over. Prior to time t, Node #0 does not have descriptors that can be recycled and so the table is resized, and the new indices are appended to the set of recycled descriptors after. After t the first descriptor, $j + 1$ can be returned to complete the registration.

References

[1] Chamberlain, Bradford L., et al. “Parallel Programmability and the Chapel Language.” IEEE International Conference on High Performance Computing Data and Analytics, vol. 21, no. 3, 2007, pp. 291–312.

[2] Chamberlain, Bradford L., et al. "Chapel Comes of Age: Making Scalable Programming Productive."

[3] https://www.cs.rochester.edu/u/ljenkin4/PGASAtomics.pdf

[4] https://github.com/chapel-lang/chapel/issues/6663

[5] Jenkins, Louis. "RCUArray: An RCU-Like Parallel-Safe Distributed Resizable Array." 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2018.

[6] Chamberlain, Bradford L. "Multiresolution languages for portable yet efficient parallel programming." (2007).

[7] https://github.com/chapel-lang/chapel/pull/8842

[8] https://summerofcode.withgoogle.com/archive/2017/projects/6530769430249472/

Intellectual Merit Criterion

Overall Assessment of Intellectual Merit
Excellent

Explanation to Applicant
Highly accomplished applicant with many publications. The proposal itself is a very detailed and fleshed-out research project.

Broader Impacts Criterion

Overall Assessment of Broader Impacts
Excellent

Explanation to Applicant
This applicant has already contributed to helping under-represented minority students enter computer science.

Summary Comments
A compelling biography makes this very impressive application even stronger.

Intellectual Merit Criterion

Overall Assessment of Intellectual Merit
Good

Explanation to Applicant
Although the applicant's academic record is not as good as that of other applicants at the same stage in their careers, the applicant has commendable research experience and has participated as a co-author in several published works. The proposed research is to develop scalable software atomics for partitioned global address space. The application is supported by very strong reference letters however, it is not very well written and does not follow the required format.

Broader Impacts Criterion

Overall Assessment of Broader Impacts
Good

Explanation to Applicant
The application does not appropriately follow the application guidelines and fails to give appropriate attention to the broader impact component. This is despite some of the references highlighting the applicant's mentoring skills.

Summary Comments
The application is good but does not appropriately follow the guidelines. The broader impact component is not appropriately addressed.

Intellectual Merit Criterion

Overall Assessment of Intellectual Merit
Fair

Explanation to Applicant
There are areas of strength in the intellectual merit of education and areas of weakness in areas that may impact the project. With proper direction, the weaknesses in certain areas may be overcome according to the information provided in the submission.

Broader Impacts Criterion

Overall Assessment of Broader Impacts
Very Good

Explanation to Applicant
The project is quite interesting and would have a very significant and positive impact.

Summary Comments
The submission demonstrates a worthwhile project although the intellectual merit is not consistent throughout with significant strengths. This appears to be able to overcome given proper mentor guidance.

Intellectual Merit Criterion

Overall Assessment of Intellectual Merit
Very Good

Explanation to Applicant
The applicant has a good academic background with several peer-reviewed publications and internship experiences. This applicant would benefit from a more explicit project plan with incremental milestones.

Broader Impacts Criterion

Overall Assessment of Broader Impacts
Excellent

Explanation to Applicant
The applicant has an inspired relationship with STEM research and expresses a path of evolution very well. The applicant's previous research provides a solid path forward with this research endeavor. The research expressed will provide several deliverables into the field.

Summary Comments
The applicant has the academic background and experience to lead this research project with minimum oversight with the ability to provide a substantial research deliverable. The applicant has several very strong letters of recommendation that speak both to academic aptitude to leadership personality. The research direction would benefit from a clearer research plan.