Please tell why you would be a good fit for …

Very long-term goals: My PhD thesis work had focused on low-overhead multi-core scheduling techniques for use in the context of scientific and high-performance computing. My long-term goal within, say, the next 10 years is to use the research I have done during my PhD to make an impact to the field of computer science through offering new perspectives to a particular area of computer science or to develop an exemplar technological solution to a problem. By being selected to attend the forum, I could have a unique opportunity to help me achieve my long-term goal in three key ways.

First, I could learn from experts their opinion about future trends in technology, which will help me to better anticipate and navigate the future trends that I would likely encounter through my career. For example, given current advances in quantum computing research, what is the potential impact of novel quantum computing technology to my PhD research topic, within the next 5 years? Could I make modifications to the techniques in my PhD thesis so that they could provide benefits in the context of a quantum computing technology? Knowing the answers to these types of questions would be beneficial to strengthening the thesis work I have done, and to apply my work in the future.

Second, I could learn how to formulate the problem in my PhD thesis mathematically. This forum is unique to me compared to conferences I have gone to up to this point, in that it brings together mathematicians and computer scientists together in one forum. By discussing and learning from experts in mathematics how to formulate my thesis research mathematically, I believe that I can more effectively apply my research to areas outside my area of specialization. An additional benefit of talking with experts in the field of mathematics is that I will be able to increase my network to mathematicians, rather than restricting my network to only computer scientists; this could allow me to potentially form new types of research collaborations, which are mathematically-oriented, in the future.

Third, and perhaps most importantly, I could understand problems in society that are impacted by my dissertation research by learning from experts how problems in society can be impacted by computer science and mathematics in general. In the context of my research, could the low-overhead multi-core scheduling techniques in my thesis be used for developing energy-efficient operating systems for laptops used in developing regions of the world? The forum’s discussions would enlighten me on how to implement this in the near-term, and learning from experts how to apply my thesis work to real research problems in society would help me apply my thesis to real-world, practical problems in industry that I may encounter throughout my profession.

Overall, learning about experts’ viewpoints will allow me to formulate a well-informed and well-rounded vision that I can use for advancing myself in the next several years of my career. The forum is also relevant to my current research directions, and near-term goals. For the above reasons, I believe I could be a good candidate for the Heidelberg Laureate Forum, and hope to be accepted to be part of the Heidelberg Laureate Forum this year.

Reference letter by Prof Cappello:

I am writing this letter in support of Vivek Kale for his application for invitation to the Heidelberg Laureate Forum. I have known Vivek Kale since I moved to University of Illinois at Urbana Champaign as a visiting professor in March 2009 to establish the INRIA-Illinois joint-laboratory on peta-scale computing at NCSA, and have had a continued relationship with Vivek throughout his PhD. Before moving to Illinois, I led a team of 30+ people with up to 7 permanent researchers, working on High Performance Parallel and Distributed Computing. In particular, I published at the ACM/IEEE Supercomputing conference in 2000 one of the first papers about hybrid MPI+OpenMP programming of cluster of multi-core processors, a novel method of implementing a large number of numerical simulations and scientific applications. I remain engaged in this area.

The MPI+OpenMP programming model is currently one of the most popular ones to use for scientific simulations intended to run at large-scale on modern supercomputers. However, although MPI+OpenMP implementations have made outstanding progress since 2000 on a large number of current supercomputers of 1,000 processors (or, nodes), managing the OpenMP threads and their workload efficiently still remains complex as scientists scale applications to 10,000 nodes or more. The problem comes from the fact that bulk synchronous execution is used by many HPC applications, due to the domain-specific constraints in the application, which cannot relax the synchrony of communication. Given this, if there are small performance fluctuations due to operating system events (which we refer to as noise, or interference), in an MPI processes within a particular time step of a simulation, that noise can amplify, producing a major performance slowdown at large scale. This problem was first established in 2003 by Petrini et al. Much work has been done to quantify and understand the impact of system noise (Tsasfir ICS05, Hoefler SC10), showing the continued and increasing importance of this problem. A second related problem is that within-node application-level imbalances are a significant factor in total imbalances across all cores of the machine, and the contribution can be larger as more cores per node exist on next-generation supercomputers.

Some low-level strategies such as co-scheduling, as suggested by Petrini, have been used to mitigate impact of system interference, but have been limited in their performance impact, and also do not handle within-node performance variations induced by non-OS events such as transient hardware variations. Additionally, within-node application-level imbalance can be handled by measurement-based load balancers such as Zoltan or Charm++, but this is inefficient and challenging to do because its mechanisms are inefficient. To address both of these problems together, Vivek followed a new strategy: develop an intelligent mixture of OpenMP static+dynamic scheduling so that the scheduler overheads could be reduced and adaptivity could be increased, both of which can help to significantly increase performance. The proposed approach consists of experimental tuning of various mixtures of static and dynamic scheduling, so that within-node imbalances can be mitigated without allowing the overhead of dynamic scheduling itself to extend the critical path of application execution. Using these scheduling strategies to balance will significantly help to reduce total application imbalance across all cores. The resulting paper that he wrote on this scheduling strategy was his first published paper and received a best paper award at the EuroMPI 2010 conference held in Stuttgart.

Vivek then continued this work towards a Ph.D. under the supervision of Bill Gropp. He first focused on his mixed static/dynamic scheduling strategies for dense matrix factorizations. Dense matrix factorizations are a class of numerical linear algebra computations that are highly time consuming in many engineering applications such as the design of an airplane wing. Vivek worked with Simplice Donfack of INRIA to improve the performance of a communication-avoiding LU code and obtained a 35% performance improvement over a standard DPLASMA implementation, which is the standard implementation for LU factorization at the time. Simplice and Vivek worked together to improve performance of these schedulers for a variety of matrix sizes through a careful tuning of the mixture of static and dynamic scheduling. Notably, his schedulers achieved higher performance than two widely /used industrial implementations of the same algorithm. These performance gains were done with the help of a performance model and theoretical analysis developed by Vivek. This is an important result since communication-avoiding LU code is already highly optimized on multi-core architectures. Vivek then published a paper with Simplice Donfack to discuss these results in IPDPS 2012. He visited INRIA France to work with Simplice (under supervision of my colleague Laura Grigori) on continued work for one month. This collaboration shows Vivek’s ability to work together with other researchers at international level and to contribute his strengths for a greater success of the team. (-- add exascale --)

Concurrently, Vivek worked on enhancing his strategy on using the measurement-based load balancing techniques used in Charm++ RTS to further guide his schedulers. This allowed performance gains on production machines such as ORNL's Cray and TACC's Ranger. This is a fundamental enhancement because it allows his scheduler using knowledge of the previous iterations to make scheduling decisions. Through the use of these techniques VIvek's scheduler incorporated an aspect of intelligence during the application's execution, which allowed for increasing performance of an application. These enhancements are particularly beneficial for bulk-synchronous simulations involviong several application timesteps. This work led to a publication of a paper in HiPC2011. This was done in collaboration with Abhinav Bhatele, at that time PhD student/postdoc working within the Charm++ group at Illinois, and now a post-doc at LLNL. This conference is highly competitive (~19% acceptance rate). Vivek presented this work at the conference in Bangalore, India in December 2011.

In January 2012, Vivek was awarded a Lawrence Scholar Fellowship, which awarded him funding from the United States Department of Energy to do research on his scheduling strategies for mitigating the amplification on DOE supercomputers. The award funding is for up to 4 years. He has worked from Lawrence Livermore Laboratory in California while doing his PhD at Illinois. His collaboration team involves Bronis de Supinski (mentor), Todd Gamblin, Torsten Hoefler, and Bill Gropp (advisor). The five of them meet for weekly conference calls, discussing ongoing progress on the implementation of his scheduling techniques. During his time at LLNL up to this point, Vivek has investigated how to make his scheduling approach usable for real-world applications. He has developed, under the supervision of Bronis de Supinski, a dynamic scheduler library integrated within OpenMP, where OpenMP uses information of the MPI runtime to possibly further guide the scheduling strategies. Specifically, each process tunes the scheduler parameters based on its properties. This work resulted in contributions of a source-to-source transformation of OpenMP codes, along with slack prediction techniques. He also addressed the problem of building an interface between two programming models within an hybrid programming model. He has developed other techniques such as using a scheduler variable task sizes, and integrated them into his software infrastructure. He along with his co-authors Bronis de Supinki, Todd Gamblin, Torsten Hoefler, and Bill Gropp has published a poster in the Supercomputing Conference in 2012 and a journal paper on dynamic scheduling techniques. They recently submitted to a top-tier high-performance computing conference the work on experimentation and development of schedulers done at LLNL, with a particular emphasis on the enhancements to the scheduler strategies done through systematic software implementation. With this, Vivek developed a scheduling library to attain additional performance benefits of pure static scheduling in his mixed static/dynamic technique, focusing on load imbalanced applications which use partitioning of work (e.g., Morton Orders for Molecular Dynamics Simulations) to improve spatial locality, which is beneficial only when using static scheduling. Vivek made this pre-determined partitioning of work still be beneficial in the context of his mixed static/dynamic scheduling strategy. This led to a paper in EuroMPI 2014 presented in Madrid, Spain in September.

Vivek also concluded his collaboration with Simplice in November when he presented a poster with Simplice at SC14 in New Orleans, in which he showed how different mixtures of scheduling strategies could be defined to handle different tradeoffs between load balance and locality. In this poster, he discussed a theoretical analysis of the work, and general scheduling strategy library for both numerical linear algebra and scientific applications.

The above elements and the global consistency of his research projects clearly demonstrate that Vivek Kale has the maturity of an excellent researcher, developing a timely and important research topic for the HPC community and exploring many of its theoretical and practical aspects, including the best way for others to use his approach.

Vivek defended his PhD last week, with his thesis dissertation focusing on low-overhead scheduling for improving performance of synchronous MPI programs. He is looking for postdoc positions to continue his research on low-overhead schedulers by integrating his schedulers with classical across-node load balancers, and to use techniques for any hybrid MPI+X models, not just MPI+OpenMP models.

In conclusion, Vivek Kale is an outstanding Ph. D. student. His research topic concerns a critical problem that needs to be solved in the near future. This is one of the fundamental problems in parallel computing and in the general field of computer science. This problem will be relevant as long as extreme-scale computing is necessary for the advancements in science and engineering. He already has established strong contacts and developed collaborations with several top researchers in the HPC domain. His work has shown to be directly beneficial and relevant in both a theoretical as well as practical setting. Based on Vivek's current work, along with his recently defined long-term goals as discussed in his letter of motivation, Vivek will benefit greatly through interactions with top scientists at the Heidelberg Laureate Forum, and also benefit from networking with other young aspiring computer scientists and mathematicians. Most importantly, I believe he will contribute to lively discussions and will go with motivation and great interest if he is invited to attend. Vivek has my highest recommendation to be a part of the Heidelberg Laureate Forum in August of this year.

What current problems within the areas of mathematics and computer science can my dissertation research provide benefit?

***Work bios:***

***Recommendation letters:***

Cover letter:

2408 High Meadow Lane

Champaign, IL 61822

March 9, 2015

D.E. Shaw Research

To Whom It May Concern:

I attended D.E. Shaw Research Information Session at University of Illinois at Urbana-Champaign, through an invite sent to me. A highlight of the session was a presentation showing how performance of a molecular dynamic simulation could be improved through the use of D.E. Shaw’s supercomputer cluster, Anton, which had hardware specifically designed to run time-consuming calculations done in molecular dynamic simulations as fast as possible.

With my work on dynamic scheduling strategies for multi-core nodes of supercomputers that I have done during my PhD, I believe I can apply the work to further improve performance of the molecular dynamics simulations being run on specialized hardware at D.E. Shaw. I also would like to publish additional work on my scheduling strategies, and I believe working at a research lab like D.E. Shaw Research would allow me to do this.

Please see my resume attached to get more information about my research experience and skill sets.

Sincerely,

Vivek Kale

C: 8 years : 40000

C++: 3 years : 12000

Python: 2 : 2500

Perl: 1: 500

VHDL: 0.5: 1000

Awards:

Lawrence Scholar Fellowship: This is a fellowship sponsored by the Lawrence Livermore National Laboratory and Department of Energy providing up to 4 years of funding for graduate research. Each year a selection from a pool of candidates consisting of PhD students is made from areas including Physics, Chemistry, Biology, and Computer Science. This fellowship award allows a PhD student to conduct their PhD research at Lawrence Livermore National Laboratory in California, with guidance for research given by their advisor and one or more mentors at Lawrence Livermore National Laboratory. This fellowship was awarded to me in November of 2011.

Notes on motivation and terminology: During execution of a scientific application on a supercomputer consisting of multi-core nodes, scientific application codes can significantly degrade in performance due to within-node load imbalances. My thesis work aims to address this through the use of low-overhead dynamic multi-core scheduling strategies, or ‘low-overhead scheduling’ for short. The basic technique of our low-overhead scheduling technique is described in reference 4 of the publications section of my CV.

My most significant accomplishment was obtaining on a multi-core node of a supercomputer a 64% performance improvement over a highly optimized (Communication-Avoiding) LU factorization multi-threaded implementation using low-overhead scheduling strategies along with a 34% performance gain over Intel’s MKL library implementation of LU factorization. The experimental results obtained were significant because LU factorization is a time-consuming numerical algorithm used in a large number of scientific application codes for engineering, such as the simulation of air flow over a plane’s wing, and so our scheduling strategy can help improve performance of those application codes. The results were also significant because they showed the potential of our scheduling strategies to improve performance of other scientific application codes. This work was published in 2012 at IPDPS, a top conference in the area of high-performance computing.

Note: The motivation for the development of each of the designs and terminology is explained in the most significant accomplishment (“Note on …”).

One design was a low-overhead scheduling library for improving performance of implementations of dense matrix factorization computations (for example, LU factorization) running on multi-core nodes of supercomputers. This size of the library was about 4000 lines of code. The code was written in C. I wrote roughly 50% of the code for the library, and the other 50% was written by my collaborator, Simplice Donfack of INRIA-Saclay. The purpose of this library was to demonstrate how dense matrix factorizations, a type of numerical algorithm frequently used in scientific application codes, could obtain performance improvements using low-overhead schedulers specifically designed for tuning performance of dense matrix factorization computations. This library also served to demonstrate how numerical linear algebra computations in general could obtain performance improvements through the use of these scheduling strategies (and the code of this library is being integrated into a large software system for numerical linear algebra computations called DPLASMA). The functionality of this library is as follows. Given a threaded dense matrix factorization code, a user adds our library’s scheduler functions before and after the threaded dense matrix factorization computation within the code. The user then experimentally determines the scheduler’s parameter value that would obtain the best performance for the code during its execution. These functions schedule work of the dense matrix factorization computation in a way that is low-overhead through the use of an intelligent blend of static and dynamic scheduling along with the use of experimental tuning of the scheduler parameters by the application programmer. The expected result of applying our library’s scheduling strategies to a dense matrix factorization code is a faster execution of that dense matrix factorization code on multi-core nodes.

Another design was a low-overhead scheduling library for scientific application codes. The size of the library is about 5500 lines of code. The library was written in C. My advisor led the project for the design of this library, and I wrote all of the code for the library. The purpose of the library was to provide our scheduling strategies generally for threaded scientific application codes, rather than specifically for threaded numerical linear algebra codes. The set of scheduling strategies used in this library are more sophisticated (though less special-purpose) than that of the library for dense matrix factorizations. Details of these scheduling strategies are in reference 6 in the publications section of the CV. The functionality is as follows. Given a threaded scientific application code implemented using MPI, an application programmer adds the library’s functions before and after the threaded computation regions of the application code. These functions are used by the runtime to schedule the work in the threaded computation region, using one of the scheduling strategies available in the library (with the application programmer getting to choose it). The user then experimentally determines the best performing scheduler’s parameter value(s). The expected result of this design is reduced application code execution time compared to that of the original application code.

A third design was a runtime software system for our scheduling strategy, which automatically adjusted the parameters of the low-overhead scheduling strategies implemented in the scheduling library described above. This size of the runtime software is about 7500 lines of code. The runtime is written in C++. Additionally, supporting application profiling was done using python. My advisor and LLNL supervisor led the project for this runtime. I wrote about 80% of the code for the runtime, and the other 20% was written in conjunction with software engineers at Lawrence Livermore National Laboratory. The purpose of the project was to provide a runtime to automatically adjust scheduler parameters for obtaining further performance improvements of application codes, and to substantially reduce application programmer effort to use our technique through automatic transformation of the application programmer’s code. The functionality is as follows. Given a threaded scientific application code using MPI, the application programmer compiles the code with our provided compiler. Our provided compiler modifies the code by placing our runtime’s function calls at the beginning and end of the threaded computation regions of the code. Then, the application programmer runs the code using our runtime software. On every invocation of the threaded computation region of the code, the function at the beginning of the region decides the best parameter values for the scheduling strategy applied in the immediately subsequent computation region. The parameter values are chosen based on history gathered through the function call invoked at the end of the computation region. The expected result is reduced application code execution time compared to that of the original application code.

Overview:

A runtime software system for our scheduling strategy, which automatically adjusted the parameters of the low-overhead scheduling strategies implemented in the scheduling library described above.

Purpose:

1. provide a runtime to automatically adjust scheduler parameters for obtaining further performance improvements of application codes.

2. substantially reduce application programmer effort to use our technique through automatic transformation of the application programmer’s code.

Functionality: