Implicitly-parallel functional dataflow for productive hybrid programming on Blue Waters

PI: Michael Wilde, University of Chicago
PI: Ioan Raicu, Illinois Institute of Technology
Application scientist: Aashish Adhikari, University of Chicago
Application scientist: Glen Hocky, Columbia University

1 Target Problem

This project explores a programming model and runtime environment which addresses the urgent yet vexing problem of how to simplify the programming of complex hybrid systems architectures.

As computing systems – from personal tablets to the largest extreme-scale systems – become increasingly parallel, and as performance gains are often made using hybrid architectures, the difficulty of programming such systems increases dramatically. The long-sought-after goal in programming model research is to make parallel programming automatic – freeing the programmer from having to explicitly code the intricate data passing and data sharing operations that are needed in common parallel programming models such as distributed and shared memory models. As hybrid architectures become increasingly turned-to as the means for further cost/performance improvements, the complexity increases with difficult orthogonal factors such as passing data across the CPU-GPU boundary.

One solution that makes parallel programming implicit rather than explicit is the dataflow model. Conceived about 35 years ago, the model comes and goes in and out of "vogue" as researchers try again to make it useful. We believe that we have successfully created a base for a programming model we call "implicitly parallel functional dataflow", as exemplified by the Swift parallel scripting language [1, 20] which is proving increasingly useful in scientific programming. This programming model has been characterized to be a perfect fit in the Many-Task Computing (MTC) paradigm.

Current software and hardware limitations prevent Many-Task Computing (MTC) workloads from leveraging hardware accelerators (NVIDIA GPUs, Intel MIC [9]) boasting Many-Core Computing architectures. Some broad application classes that fit the MTC paradigm are workflows, MapReduce, high-throughput computing, and a subset of high-performance computing. MTC emphasizes using many computing resources over short periods of time to accomplish many computational tasks (i.e. including both dependent and independent tasks), where the primary metrics are measured in seconds. MTC has already proven successful in Grid Computing and Supercomputing on MIMD architectures, but the SIMD architectures of today's accelerators pose many challenges in the efficient support of MTC workloads on accelerators. This work aims to address the programmability gap between MTC and accelerators, through an innovative middleware (GeMTC [7, 8, 10]) that will enable MIMD workloads to run on SIMD architectures. This work will enable a broader class of applications to leverage the growing number of accelerated high-end computing systems.

This project specifically addresses the following research problems:

- Programmability of hybrid architectures (both within nodes, multi-core and many-core, and across nodes)
- Supporting a diverse set of accelerators, from NVIDIA GPUs, AMD GPUs & APUs, and Intel MIC
- Scaling downwards (e.g. increasing the spectrum of applications by decreasing leaf-function granularity)
- Scaling upwards (increasing scalability to multi-petaflop supercomputers)
- Language interoperability (integrate with many languages and programming models for performing leaf tasks: C/C++, Fortran, UPC, MPI, Chappel/X10/etc, global arrays)
- Runtime facilities for tracing and debugging hybrid applications at extreme scale
- Evaluation of the programming model on applications in biophysical dynamics and the theoretical chemistry of glass-state materials.

We believe the Blue Waters supercomputer is an ideal testbed to explore a petascale hybrid architecture at extreme scale with 3K GPUs and up tp 300K floating point cores on a real diverse set of scientific applications. This proposal satisfies two specific categories, namely 1) Scaling Studies and 2) Proposals from non-traditional communities.

We have scaled up the Swift parallel programming language up to 131,072 cores on a BlueGene/P [19] supercomputer. In addition, Swift is in constant use on the UChicago Cray XE6 system "Beagle", with 17K cores. We thus expect to be able to scale Swift readily to 3K nodes and 3K GPUs. The GeMTC framework has only been tested in small testbeds of 12 GPUs, and we currently lack a larger testbed to explore its performance and scalability. Getting access to Blue Waters with 3K GPUs would be an invaluable computer science research opportunity to explore the GeMTC framework's scalability in combination with the proven Swift system.

We believe that this work represents a non-traditional community for HPC systems in general and Blue Waters in particular, one that focuses on Many-Task Computing. MTC has been applied to clusters [16], grids [15], clouds [13], and even some supercomputing [17]. However, supercomputing has been predominantly supporting HPC applications. However, there are many advantages to MTC, such as improved programmability, implicit parallelism, and improved fault tolerance – all reasons why some applications and researchers in the community have adopted MTC as their programming model for petascale supercomputers [2, 4, 18, 20, 31, 32]. Given the popularity of hybrid supercomputer architectures, being able to showcase the use of MTC in hybrid architectures such as Blue Waters, at petascale levels, is a critical activity towards the acceptance of MTC as a viable programming model for future supercomputers.

2 Team Members

- **Michael Wilde**
  - Software Architect, Math and Computer Science Division, Argonne National Laboratory
  - Fellow, Computation Institute, University of Chicago
  - Contact: 1-630.252.7497; wilde@mcs.anl.gov
- **Ioan Raicu**
  - Assistant Professor, Computer Science, Illinois Institute of Technology
  - Guest Research Faculty, Math and Computer Science Division, Argonne National Laboratory
  - Contact: 1-847-722-0876; iraicu@cs.iit.edu
- **Aashish Adhikari**
  - Graduate Student, Chemistry, University of Chicago (PhD expected Dec. 2012)
  - Postdoctoral research associate, University of Chicago, expected start Jan 2013
  - Contact: 1-413-884-2993; aashish@uchicago.edu
- **Glen Hocky**
  - Graduate student, Chemistry, Columbia University
  - Research assistant, David Reichmann Lab, Columbia
  - Contact: 1-646-571-8552; hockyg@gmail.com

3 Description of Systems Software and Application Codes

3.1 *Swift implicitly parallel functional dataflow language*

This research involves pursuing the integration between data-flow driven parallel programming systems (e.g. Swift 1, 4, 20) and hardware accelerators (e.g. NVIDIA GPUs, AMD GPUs, and the Intel MIC). Swift has been successfully used in many large-scale computing applications.[1, 2, 3, 4, 5, 6, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33] The scientific community has adopted Swift as a great way to increase productivity in running complex applications via a dataflow driven programming model, which intrinsically allows implicit parallelism to be harnessed based on data access patterns and dependencies. Swift is a parallel programming system that is encompassed by the MTC programming model, and has been shown to run well on tens of thousands of nodes with task graphs in the range of millions of tasks.[31]

3.2 *GeMTC*

Current software and hardware limitations prevent Many-Task Computing (MTC) workloads from leveraging hardware accelerators (NVIDIA GPUs, Intel MIC) boasting Many-Core Computing architectures. MTC has already proven successful in Grid Computing and Supercomputing on MIMD architectures, but the SIMD architectures of today's accelerators pose many challenges in the efficient support of MTC workloads on accelerators. This work aims to address the programmability gap between MTC and accelerators, through an innovative middleware that will enable MIMD workloads to run on SIMD architectures. This work will enable a broader class of applications to leverage the growing number of accelerated high-end computing systems, such as Blue Waters.

Through the use of the GeMTC [7, 8, 10] framework, this work aims to enable Swift to efficiently use accelerators (such as NVIDIA GPUs and Intel MIC) to further accelerate a wide range of applications, on a growing portion of high-end systems. GPUs are one of the most effective ways to provide acceleration on HPC resources. However, a programmability gap still exists between applications and accelerators. Researchers and developers are forced to work within the constraints of closed environments such as the CUDA GPU Programming Framework [11, 12].

The CUDA framework can only support 16 concurrent kernels, one kernel per streaming multiprocessor (SM). [12] Besides the limited number of concurrent kernel executions, all kernels must start and end at the same time, causing extreme inefficiencies in heterogeneous workloads. The GeMTC framework operates at a finer granularity level, at the warp level, between cores and SMs. The GeMTC framework can trade local memory for increased concurrency (up to 192 concurrent micro-kernels [10]). GeMTC allows independent micro-kernels (essentially MIMD) to be launched and managed on NVIDIA GPUs (which have SIMD architecture). Our preliminary results in the costs associated with managing and launching concurrent micro-kernels on NVIDIA Kepler GPUs show that our framework is able to achieve high efficiency (e.g. 95%+) for heterogeneous MTC workloads with task granularities of milliseconds. Essentially, the GeMTC framework has effectively changed the programming model of NVIDIA GPUs from SIMD to MIMD, potentially opening the door to a larger class of non-traditional GPU applications.

We expect our results to be applicable to the many HPC resources where GPUs are now common, especially BlueWaters. Finally, we plan to explore applications from different domains such as medical imaging, economics, astronomy, bioinformatics, physics, and many more. We will continue to push the performance envelope by enabling many MTC applications and systems to leverage the growing number of accelerated high-end computing systems. We also expect this work to enable other classes of applications to leverage accelerators, such as MapReduce and ensemble MPI. We hope to influence future accelerator architectures by highlighting the need for hardware support for MIMD workloads.

The GeMTC framework is spread over both a host CPU and accelerator. We have developed a GeMTC API, written in C, that allows parallel programming languages to send the framework workloads of tasks, and receive results back on a per task basis. Essentially, the GeMTC Super-Kernel runs a pilot kernel, keeping the GPU in a busy running wait state, waiting for MTC tasks to arrive and be processed. These tasks can be traditional CUDA kernerls (named micro-kernels due to them being started from the super-kernel) that utilize a large portion of the GPU, or they can be MTC micro-kernels that run on a minimal number of cores.

Another contribution of the GeMTC framework is an advanced dynamic memory management system, which could be useful to executing MTC workloads on GPUs, but also for executing traditional CUDA applications on GPUs. When conducting initial testing of MTC workloads on accelerators we discovered that the default memory management system available in CUDA was not designed for dynamic memory management. We developed our own version of cuda_malloc(), referred to as gemtc_malloc(). Our memory allocation is nearly 8X faster (14 microseconds compared to 110 microseconds), and it is stable with increasing number of memory allocations with more than 1000X faster performance than the traditional memory management after tens of thousands of moeory allocations. The GeMTC framework can operate at speeds of 14K tasks per second (performing sleep 0 tasks), making it extremely light-weight. The Swift/T parallel programming system has been measured to operate at 1K tasks/sec per node. We anticipate that when combining Swift/T and GeMTC, we will be able to achieve high efficiency of the underlying hardware with 100ms tasks or longer, a much finer granularity than what is currently possible without support for accelerators.

3.3 Glass material modeling application

Many recent theoretical chemistry studies of the glass transition in model systems have focused on calculating from theory or simulation what is known as the mosaic length. Glen Hocky of the Reichman Group at Columbia is evaluating a new cavity method for measuring this length scale, where particles are simulated by molecular dynamics or Monte Carlo methods within cavities having amorphous boundary conditions.

In this method, various correlation functions are calculated at the interior of cavities of varying sizes and averaged over many independent simulations to determine a thermodynamic length. Hocky has already used this method to investigate the differences between three glass systems that all have the same structure but differ in subtle ways and has determined that this thermodynamic length can distinguish the variations among the three systems. Moreover, his results showed for the first time a direct connection between point-to-set length and the dynamical behavior in the glassy systems [ref: Hocky-PRL2012].

In order to achieve these results, the glass cavity application performs 100,000 Monte Carlo steps in 1 – 2 hours. Jobs of this length are run in succession and strung together to make longer simulations tractable. The input data to each simulation is a file of about 150 KB representing initial glass structures and a 4 KB file describing which particles are in the cavity. Each simulation returns three new structures of 150 KB each, a 50 KB log file, and the same 4 KB file describing which particles are in the cavity.

Each script run covers a simulation space of 7 radii by 27 centers by 10 models, requiring 1,890 jobs per run. Three model systems are investigated for a total of 90 runs. Swift mappers enable metadata describing these aspects to be encoded in the data files of the simulation campaigns to assist in managing the large volume of file data. Hocky used four Swift scripts in glass simulation campaigns. The first, **glassCreate**, generates an equilibrated configuration at some temperature; **glassAnneal** takes those structures and lowers the temperature to some specified temperature; **glassEquil** freezes particles outside a spherical cavity and runs short simulations for particles inside; and the script **glassRun**, described below, is the same but starts from equilibrated cavities. The simulation is restartable in one- to two-hour increments; it works together with the Swift script to create a simple but powerful mechanism for managing checkpoint/restart across a long-running, large-scale simulation campaign.

The **GlassRun** function starts by extracting a large set of science parameters from the Swift command line. It uses the built-in function **readData** to read prepared lists of molecular radii and centroids from parameter files to define the primary physical dimensions of the simulation space. A selectable energy function to be used by the simulation application is specified as a parameter. The script leverages Swift flexible dynamic arrays to create a 3D array for input and a 4D array of structures for outputs. The entire body of **GlassRun** is a four-level nesting of Swift parallel foreach statements. These loops perform a parallel parameter sweep over all combinations of radius, centroid, model, and job number within the simulation space. A single run of the script immediately expands to an independent parallel invocation of the simulation application for each point in the space: 1,890 jobs for the minimum case of a 7 x 27 x 10 x 1 space. In prior work, the fourth dimension of the simulation space was fixed at one. This value can be increased to define subconfigurations that perform better Monte Carlo averaging, with a multiplicative increase in the number of jobs.

The advantages of managing a simulation campaign in this manner are borne out well by Hocky's experience: the expression of the campaign is a well-structured, high-level script, devoid of details about file naming, synchronization of parallel tasks, location and state of remote computing resources, or explicit data transfer. Further studies are of great interest, some of which require re-executing the entire campaign for different model systems. The fact that these simulations were driven by location-independent Swift scripts will enabling Hocky to do so on Blue Waters under Swift with relative ease. Other studies are planned that require introducing new simulation techniques into the core glassRun code, but organizing new simulations across many high performance resources will require only minor tweaks to the robust workflow framework already in place.

3.4 Protein structure application

As protein targets increase in size, the need for petascale computing becomes increasingly important to model . Current prediction methods have limited accuracy even for proteins on the order of 100 residues when homology-based information is minimal. To fold larger and multi-domain proteins, statistical sampling becomes a limiting factor. Additionally, current docking algorithms are inconsistent. The most glaring issue in the folding of multi-domain proteins and the docking of complexes is "induced-fit", wherein each of the constituents (either the individual domains or the binding partners) change shape and can no longer be treated as rigid entities. As a result, the folding and docking processes require extra sampling rounds, which greatly increases the amount of computation power required. Furthermore, this entire class of calculations generally are going to require a much larger number of loosely coupled calculations rather than a few massively parallel calculations.

The identification of the protein sequences in a genome has transformed biological studies. But it is only the starting point. Amino acid sequence codes for structure, which often determines function. At the next level, protein association is integral to signaling networks which control and orchestrate cellular processes. Our longterm goal is the ability to take a set of genes identified in a biological process and provide the structure and function of the proteins, as well as identify their interactions and signaling connections, including the positive and negative feedback one interaction has on another. Petascale computation is critical for this far-reaching goal.

OOPS – the Open Protein Simulator – is a suite of C++ programs for the prediction of the structure of proteins with minimal use of information derived from sequence similarity or homology to other proteins. OOPS derives its speed and accuracy from the use of a “C_β” model, an accurate statistical potential, and a search strategy involving iterative fixing of structure in multiple “rounds” of folding. Since OOPS uses minimal homology information and a reduced representation, its success depends crucially on describing the “protein physics” correctly. Great effort has been devoted to the energy function, e.g., interactions are conditional on backbone geometry and the relative orientation of side chains. In 2009, its accuracy exceeded available all-atom potentials, and it has been significantly improved since then. Our homology-free 2^o and 3^o structure predictions for small proteins rival or exceed homology-based methods with (expensive) explicit side chains, engendering optimism for continued progress.

We OOPS is currently executed in parallel on modest computing clusters using an ad-hoc Python script that submits independent parallel jobs, analyzes results, and orchestrates the iteration of a parallel solution to a folding problem. We propose to adapt OOPS to effectively utilize the hybrid architecture of the Blue Waters XK6 nodes, thus enabling OOPS to be applied to challenging folding targets and providing the requisite number of docking simulations (10^4 – 10^6) of challenging binding targets. OOPS’s performance, accuracy, capabilities and usability will be enhanced and will benefit from live scientific usage and testing on this petascale computing platform.

4 Porting, Testing, and Scalability Plans

This work requires 6M core-hours, or the equivalent of 375K node hours, approximately 60% of which will require GPU nodes. This would allow us enough resources to ensure Swift scales well on Blue Waters (we do not anticipate any issues given that we have scaled it to BlueGene/P scales at 32K nodes/128K cores). This would also allow us to scale the GeMTC framework up to 3K GPU scales. Due to the higher computing capacity of GPUs compared to CPUs, the successful running of GeMTC at 3K GPU scales might produce unprecedented loads on the Swift system, in terms of tasks/sec and data-management requirements. We have the needed expertise to tackle both the potential scheduling and storage challenges that we might encounter as we scale up Swift/GeMTC. Through the UChicago relationship with Cray, we have ready access to Cray test systems in addition to the UChicago Beagle system, and more importantly to Cray support staff and OS, compiler, and hardware specialists.

5 Resources Required

We expect to use 20% of our resources for large scale testing, 20% for application evaluation of protein structure prediction, and 40% for application evaluation of glass materials simulation. The estimated resource needs are: 6M hours total, 2.4M core hours on standard nodes (150K node hours XE6 nodes), 3.6M cores hours on GPU nodes (225K node hours XK6 GPU nodes).

6 Support Plan

The PIs involved with this work have extensive experience on petascale supercomputers, such as the IBM BlueGene/P [19] and Q, as well as the Cray XE6. The PIs have participated in INCITE proposals, and have successfully run Swift at full 40K-node scales on the BlueGene/P supercomputer. This work also involves NVIDIA specific work to enable MTC workloads on GPU accelerators. We have a functional prototype GeMTC and have tested it at 12-node scales on a GPU cluster at IIT. We expect to scale up the Swift/GeMTC framework to the full 3K-node/GPU scale of Blue Waters.

7 Sources of Funding

This work is in part supported by the NSF OCI CAREER award (1054974) – “Avoiding Achilles’ Heel in Exascale Computing with Distributed File Systems”, the NSF OCI award (1148443), “SI2-SSE: Enhancement and support of Swift parallel scripting” and the DOE X-STACK award (FWP-57810) – ”ExM: Systems support for extreme-scale many-task applications”.

8 Additional Required Documents

MICHAEL J. WILDE – BIOGRAPHICAL SUMMARY

Education and training

University of Illinois, Urbana, Illinois	Computer Science	B.S., 1977

Research and professional experience

2006 – 2009	Education, Outreach and Training Coordinator, Open Science Grid
2004 – present	Fellow, Computation Institute, The University of Chicago
2001 – present	Software Architect, MCS Division, Argonne National Laboratory, Argonne, IL.
2001 – 2006	Project Coordinator, Grid Physics Network (NSF GriPhyN ITR-0086044)
1996 – 2000	Chief Technology Officer, CoolSavings.com Inc., Chicago, IL
1990 – 1996	Manager, New Product Development, Computer Associates, Lisle, IL
1985 –1986	Technical Project Manager, Lachman Associates, Westmont, IL
1981 – 1985, 86-90	Consulting developer to AT&T Bell Laboratories, Murray Hill NJ & Naperville IL
1977 – 1981	Consultant and Systems Programmer, Univ. of Illinois and Univ. of Toronto

Publications directly related to current proposal

1. Wilde, M., Hategan, M., Wozniak, J., Clifford, B., Katz, D. S., Foster, I. Swift: A Language for Distributed Parallel scripting, Parallel Computing, vol 37, issue 9, pp 633-652, Sep. 2011.
2. Wilde, M., Foster, I. Iskra, K., Beckman, P., Zhang, Z., Espinosa, A., Hategan, M., Clifford, B., Raicu, I., Parallel Scripting for Applications at the Petascale and Beyond," IEEE Computer 42, no. 11, 4-5, Nov. 2009.
3. Debartolo, J., Hocky, G., Wilde, M., Xu, J., Freed, K. F. & Sosnick, T. R. Protein Structure Prediction Enhanced with Evolutionary Diversity: SPEED. Protein Sci., published online Jan. 11, 2010.
4. Raicu, I., Zhang, Z., Wilde, M., Foster, I., Beckman, P., Iskra, K. and Clifford, B. Toward Loosely Coupled Programming on Petascale Systems, 2008 ACM/IEEE Conference on Supercomputing, 2008.
5. Kenny, S, M. Andric, M. Wilde, Michael C. M. Neale, S. Boker, S. L. Small, Parallel Workflows for Data-Driven Structural Equation Modeling in Functional Neuroimaging. Frontiers in Neuroscience, vol. 3, September 2009. *doi: 10.3389/neuro.11.034.2009*

Additional publications related to current proposal

6. Small, S. L., Wilde, M., Kenny, S., Andric, M., & Hasson, U. (2009). Database-Managed Grid-enabled Analysis of Neuroimaging Data: The CNARI Framework. International Journal of Psychophysiology 73, no. 1, 62–72, 2009.
7. Hasson, U., Jeremy, J., Wilde, M., Nusbaum, H., Small, S. Improving the Analysis, Storage and Sharing of Neuroimaging Data Using Relational Databases and Distributed Computing. NeuroImage 39, no. 2, 693–706, 2008. PMC2169517.
8. Zhao, Y., Wilde, M. and Foster, I. Applying the Virtual Data Provenance Model International Provenance and Annotation Workshop, Chicago, pp. 148–161, 2006.
9. Clifford, B., Foster, I., Voeckler, J., Wilde, M. and Zhao, Y. Tracking Provenance in a Virtual Data Grid. Journal of Concurrency and Computation, Practice and Experience 20, no. 5, 565–575, Aug. 2007.
10. Stef-Praun, T., Clifford, B., Foster, I., Hasson, U., Hategan, M., Small, S., Wilde, M and Zhao,Y. Accelerating Medical Research Using the Swift Workflow System. In Studies in Health Technology and Informatics, edited by N. Jacq et al., volume 126, 2007.

Synergistic Activities

1. *PI*, NSF SDCI grant OCI- 0721939 “Improvement of the Swift Science and Engineering Workflow Environment”, grant OCI-0944332, “Design and evaluation of a programming model for petascale parallel scripting”, and DOE ASCR grant ExM: systems support for extreme scale many task applications; *PI*, University of Chicago, Computation Institute PI for workflow component of caGrid for the Cancer Biomedical Informatics Grid (caBIG, 2006-2007).
2. *Analytics Challenge Award*, Supercomputing 2007 for collaboration on project “Angle: detecting anomalies and divergent behavior from distributed data in near real time,” http://www.ncdm.uic.edu/gfx/sc/sc2007-HPC-Award.jpeg (team award)
3. *Co-organizer and program committee member*, International Provenance and Annotation Workshop, 2006, Chicago, Illinois.
4. *Project coordinator*, NSF Large ITR “GriPhyN – The Grid Physics Network”, 2001-2006.
5. *Education coordinator*, DOE/NSF Open Science Grid, 2004-2006.

Collaborators and Other Affiliations

i. Collaborators and co-authors

Rachana Ananthakrishnan(UChicago), James Annis(Fermilab), Paul Avery(UFlorida), Marjorie Bardeen(Fermilab), Peter Beckman (Argonne), Bennett Bertenthal(IndianaU), Alina Bejan(UChicago), Thomas Binkowski(Argonne), Kent Blackburn(Caltech), Steven Boker(Virginia), Richard Cavanaugh(UFlorida), Yue Chen(UChicago), Benjamin Clifford(UChicago), Jed Dobson(Dartmouth), Joseph Debartolo(UChicago), Ewa Deelman(USC), Mark D'Souza(Argonne), Catalin Dumitrescu(Fermilab), James Evans(UChicago), Wu Fang(USouthampton), Ian Foster(UChicago), Karl Freed(UChicago), Jaime Frey(UWisconsin), Robert Gardner(UChicago), Maryellen Giger(UChicago), Yolanda Gil(USC), Eric Gilbert(UIUC), Roscoe Giles(BostonU), Robert Grossman(UChicago), Uri Hasson(Trento), Mihael Hategan(UChicago), Glen Hocky(Columbia), Donald Holmgren(Fermilab), Mark Hereld(UChicago), Kamil Iskhra (Argonne), Andrew Jamieson(UChicago), Andrzej Joachimiak(Argonne), Thomas Jordan(Florida), Daniel Katz(UChicago) Carl Kesselman(USC), Michael Kubal(UChicago), Scott Lathrop(UChicago), Miron Livny(UWisconsin), Patrick McConnell(Duke), Greg Malewicz(Google), Natalia Maltsev(UChicago), Marta Mattoso(UFRJ), Gurang Mehta(USC), Luiz Meyer(UFRJ), Luc Moreau(USouthampton), Michael Neale(VT), Howard Nusbaum(UChicago), Scott Oster(OSU), Michael Papka(UChicago), Ruth Pordes(Fermilab), Elizabeth Quigg(Fermilab), Ioan Raicu(Northwestern), Alex Ramirez(HACU), Arnold Rosenberg(UMass), Allan Roy(UWisconsin), Alex Rodriguez(UChicago), Benoit Roux(UChicago), Randall Ruchti(Notre Dame), Douglas Scheftner(UChicago), Devleena Shivakumar(Schrödinger), Jonathan Silverstein(UChicago), James Simone(Fermilab), Jeremy Skipper(Cornell), Steven Small(UCI), Tobin Sosnick(UChicago), Tiberiu Stef-Praun(UChicago), Rick Stevens(UChicago), Mark Subba-Rao(UChicago), Deena Sulakhe(UChicago), Xian-he Sun(IIT), Alex Szalay(JHU), James van Horn(UCLA), Gregor von Laszewski(Indiana), Jens Voeckler(ISI), Gregory Voth (UChicago), Justin Wozniac(Argonne), Jinbo Xu(TTI-C), Yingming Zhao(UChicago), Yong Zhao(Microsoft),

ii. Graduate and postdoctoral advisors: n/a

iii. Thesis Advisees, Post–Doctoral Scholars Sponsored (Past 5 years): Postdoc: Ketan Maheshwari (Argonne), Justin Wozniak (Argonne). Graduate student: Timothy Armstrong (UChicago), Allan Espinosa (UChicago), Luiz Gadhela (UFRJ), Luiz Meyer(UFRJ), Zhao Zhang (UChicago), Yong Zhao(UChicago). Total Thesis Advisees (informal): 6, Post–Doctoral Scholars Sponsored: 2.

8.1 Ioan Raicu

8.1.1 Professional Preparation

Wayne State University	Computer Science (CS)	Bachelor of Science, 2000
Wayne State University	CS	Master of Science, 2002
University of Chicago	CS	Master of Science, 2005
University of Chicago	CS	Ph.D., 2009
Northwestern University	EECS	Postdoc, 2009 - 2010

8.1.2 Appointments

2010 - Present	Assistant Professor, Illinois Institute of Technology, CS Dept., Chicago, IL
2011 - Present	Guest Research Faculty, Argonne National Laboratory, MCS, Argonne, IL
2009 - 2010	Computing Innovation Fellow, Northwestern Univ., EECS Dept., Evanston, IL
Summer 2009	Research Visitor, NASA, Ames Research Center, NAS, Moffett Field, CA
2003 - 2009	Teaching/Research Assistant, Univ. of Chicago, CS Dept., Chicago, IL
Summer 2005/2006	Researcher (Internship), Argonne National Lab., MCS, Argonne, IL
Summer 2003	Researcher (Internship), Sun Microsystems, Sun Labs, Menlo Park, CA
2002 - 2003	Teaching Assistant, Purdue University, CS Dept., West Lafayette, IN
Summer 2002	Adjunct Assistant Professor, Univ. of Michigan, CIS Dept., Dearborn, MI
2000 - 2002	Teaching/Research Assistant, Wayne State Univ., CS Dept., Detroit, MI
Summer 2001	Researcher (Internship), Accenture Technology Labs, Palo Alto, CA
Summer 1999	System Analyst (Internship), Ford Motor Company, Dearborn, MI
1997 - 2001	Owner, High Teck Computers, Westland, MI

8.1.3 Publications

Publications most closely related to the proposed project (http://www.cs.iit.edu/~iraicu/research/publications/):

1. **Ioan Raicu**, Pete Beckman, Ian Foster. "*Making a Case for Distributed File Systems at Exascale*", ACM Large-scale System and Application Performance (**LSAP**), 2011; (5 citations)
2. **Ioan Raicu**, Ian Foster, Yong Zhao, Philip Little, Christopher Moretti, Amitabh Chaudhary, Douglas Thain. "*The Quest for Scalable Support of Data Intensive Workloads in Distributed Systems*", ACM Symp. on High Performance Distributed Computing (**HPDC**) 2009; (20 citations)
3. **Ioan Raicu**, Ian Foster, Yong Zhao. "*Many-Task Computing for Grids and Supercomputers*", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (**MTAGS**) 2008; (91 citations)
4. Yong Zhao, Mihael Hategan, Ben Clifford, Ian Foster, Gregor von Laszewski, **Ioan Raicu**, Tibi Stef-Praun, Mike Wilde. "*Swift: Fast, Reliable, Loosely Coupled Parallel Computation*", IEEE Workshop on Scientific Workflows (**SWF**) 2007; (202 citations)
5. **Ioan Raicu**, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde. "*Falkon: a Fast and Light-weight tasK executiON framework*", IEEE/ACM **SuperComputing/SC**, 2007; (177 citations)

Other significant publications:

6. Michael Wilde, Ian Foster, Kamil Iskra, Pete Beckman, Zhao Zhang, Allan Espinosa, Mihael Hategan, Ben Clifford, **Ioan Raicu**. "*Parallel Scripting for Applications at the Petascale and Beyond*", **IEEE Computer** Nov. 2009, SI on Extreme Scale Computing, 2009 (51 citations)
7. Ian Foster, Yong Zhao, **Ioan Raicu**, Shiyong Lu. "*Cloud Computing and Grid Computing 360-Degree Compared*", IEEE Grid Computing Environments (**GCE**) 2008; (835 citations)
8. **Ioan Raicu**, Zhao Zhang, Mike Wilde, Ian Foster, Pete Beckman, Kamil Iskra, Ben Clifford. "*Towards Loosely-Coupled Programming on Petascale Systems*", IEEE/ACM **SuperComputing/SC** 2008; (55 citations)
9. Jennifer M. Schopf, **Ioan Raicu**, Laura Pearlman, Neill Miller, Carl Kesselman, Ian Foster, Mike D'Arcy. "*Monitoring and Discovery in a Web Services Framework: Functionality and Performance of Globus Toolkit MDS4*", Technical Report, Argonne National Laboratory, MCS Preprint #ANL/MCS-P1315-0106, January 2006 (167 citations)
10. William Allcock, John Bresnahan, Rajkumar Kettimuthu, Michael Link, Catalin Dumitrescu, **Ioan Raicu**, Ian Foster, "*The Globus Striped GridFTP Framework and Server*," IEEE/ACM **Supercomputing/SC**, 2005; (374 citations)

8.1.4 Synergistic Activities

1. **New courses designed and taught at Illinois Institute of Technology:**
    a. Data-Intensive Computing (CS554)
    b. Cloud Computing (CS553)
    c. Introduction to Distributed Systems (CS495)
2. **Founder and chair of various workshops:**
    a. IEEE Workshop on Data-Intensive Computing in the Clouds (**DataCloud, DataCloud-SC11**) 2011, 2012; co-located with IEEE IPDPS 2011 and IEEE/ACM SC 2011/2012; co-chairs Tevfik Kosar, Roger Barga
    b. ACM Workshop on Scientific Cloud Computing (**ScienceCloud**) 2010, 2011, co-located with ACM HPDC 2010, 2011; co-chairs Pete Beckman, Ian Foster, Yogesh Simmhan
    c. ACM/IEEE Workshop on Many-Task Computing on Grids and Supercomputers (**MTAGS**), 2008, 2009, 2010, 2011, 2012 co-located with ACM/IEEE SC; co-chairs Ian Foster, Yong Zhao, Justin Wozniak
3. **Guest Editor:**
    a. Springer Journal of Grid Computing (**JOGC**), Special Issue on Data Intensive Computing in the Clouds, Springer, 2012; guest editors, Tevfik Kosar
    b. IEEE Transactions on Parallel and Distributed Systems (**TPDS**), Special Issue on Many-Task Computing, 2011; guest editors, Ian Foster, Yong Zhao
    c. Scientific Programming Journal (**SPJ**), Special Issue on Science-driven Cloud Computing, 2011; guest editors, Ivona Brandic and Ewa Deelman
4. Recent **board/steering/organizing committee member** of CCGrid14, TCSC12, HPDC12, eScience12, CCGrid12, CloudFlow12, ICAC12, JoCCASA, HPDC/SigMetrics11, CCA11, GRID11, HPDC11, HPDC10, SWF10, DSLW09, MegaJobs08, DSLW08, DSLW07, & DSLW06.
5. Recent **program committee member** of SC13, HPDC13, CCGrid13, SC12, HPDC12, CCGrid12, FutureTech12, HCW12, ICAC12, HPDC11, CLOUD11, eScience11, Grid11, TG11, GCE11, DADC11, HCW11, ISPA11, SWF11, DHCS11, SC-PhD11, HPDC10, CloudCom10, eScience10, TG10, DADC10, GCE10, SWF10, SC-PhD10, HCW10, KDCloud10, CSE10, NBiS10, TG09, GCE09, DIDC09, SWF09, CSE09, SC-PhD09, GCE08, GCE07; recent **journal reviewer** for JPDC11, TPDS11, FGCS11, IC10, FGCS10, JPDC10, TSC10, TC10, CCPE10, TC09, TPDS09, IJBPIM09, CCPE09, IJCA09, IC07, GRID07, IC07, SC06, TC06, CCPE06, and CL05.

8.1.5 Collaborators and Other Affiliations (Since 2008)

Collaborators: ***Atilla S. Balkir*** (UChicago), ***Roger Barga*** (MSR), ***Pete Beckman*** (UChicago/ANL), ***Ivona Brandic*** (VUT), ***Amitabh Chaudhary*** (ND), ***Ben Clifford*** (Consultant), ***Mary Cummane*** (Perspectives), ***Ewa Deelman*** (USC), ***Catalin Dumitrescu*** (FNAL), ***Allan Espinosa*** (UChicago), ***Xubo Fei*** (WSU), ***Jacob Furst*** (DePaul), ***Gabriele Garzoglio*** (FNAL), ***Chris Gladwin*** (Cleversafe), ***Mihael Hategan*** (UCI), ***Kamil Iskra*** (ANL), ***Sarah Kenny*** (UChicago), ***Carl Kesselman*** (USC), ***Tevfik Kosar*** (SUNY), ***Philip Little*** (ND), ***Shiyong Lu*** (WSU), ***Arthur Barney Maccabe*** (ORNL), ***Tanu Malik*** (UChicago), ***John McGee*** (RENCI), ***Christopher Moretti*** (ND), ***Veronika Nefedova*** (ANL), ***Quan T. Pham*** (UChicago), ***Marlon Pierce*** (IU), ***Ruth Pordes*** (FNAL), ***Dick Repasky*** (IU), ***Matei Ripeanu*** (UBC), ***Rob Ross*** (ANL), ***Yogesh Simmhan*** (USC), ***Marc Snir*** (ANL), ***Rick Stevens*** (UChicago/ANL), ***Xian-He Sun*** (IIT), ***Alex Szalay*** (JHU), ***Douglas Thain*** (ND), ***Jing Tie*** (UChicago), ***Gabriela Turcu*** (UChicago), ***Michael Wilde*** (UChicago/ANL), ***Justin M. Wozniak*** (ANL), ***Xi Yang*** (IIT), ***Zhao Zhang*** (UChicago), ***Yong Zhao*** (UESTC)

Graduate Advisors and Postdoctoral Sponsors: ***Alok Choudhary*** (Northwestern), ***Ian T. Foster*** (UChicago/ANL), ***Douglas Comer*** (Purdue), ***Sherali Zeadally*** (UDC)

Thesis Advisor and Postgraduate-Scholar Sponsor: ***Tonglin Li*** (IIT), ***Ke Wang*** (IIT), ***Iman Sadooghi*** (IIT), ***Scott Krieder*** (IIT), ***Dongfang Zhao*** (IIT), ***Xiaobing Zhou*** (IIT), ***Anupam Rajendran*** (IIT), ***Chen Shou*** (IIT), ***Sanjeev Gupta*** (IIT), ***Benjamin Grimmer*** (IIT), ***Zhangjie Ma*** (IIT), **Da Zhang** (IIT), ***Pedro Alvarez-Tabio*** (IIT), ***Corentin Debains*** (IIT), ***Kevin Brandstatter*** (IIT), ***Antonio Perez De Tejada*** (IIT), **Ke Yue** (IIT), ***Raman Verma*** (IIT), ***Xi Duan*** (IIT), ***Juan Carlos Hernández Munuera*** (IIT), ***Hui Jin*** (IIT), ***Wei Tang*** (IIT), ***Yuchi Tsao*** (IIT), ***Kyle Chard*** (Victoria)**,** ***Bin Zhang*** (DePaul)

9 References

[1] Yong Zhao, Mihael Hategan, Ben Clifford, Ian Foster, Gregor von Laszewski, Ioan Raicu, Tibi Stef-Praun, Mike Wilde. “Swift: Fast, Reliable, Loosely Coupled Parallel Computation”, IEEE Workshop on Scientific Workflows (SWF07) 2007

[2] Ioan Raicu, Ian Foster, Yong Zhao. “Many-Task Computing for Grids and Supercomputers”, IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08), 2008

[3] Ioan Raicu, Zhao Zhang, Mike Wilde, Ian Foster, Pete Beckman, Kamil Iskra, Ben Clifford. “Towards Loosely-Coupled Programming on Petascale Systems”, IEEE/ACM SC 2008

[4] Michael Wilde, Ian Foster, Kamil Iskra, Pete Beckman, Zhao Zhang, Allan Espinosa, Mihael Hategan, Ben Clifford, Ioan Raicu. "Parallel Scripting for Applications at the Petascale and Beyond", IEEE Computer Nov. 2009 Special Issue on Extreme Scale Computing, 2009

[5] Yong Zhao, Ioan Raicu, Ian Foster. “Scientific Workflow Systems for 21st Century e-Science, New Bottle or New Wine?”, IEEE Workshop on Scientific Workflows (SWF08) 2008

[6] Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde. “Falkon: a Fast and Light-weight tasK executiON framework”, IEEE/ACM SuperComputing/SC, 2007

[7] Scott Krieder, Ioan Raicu. “Towards the Support for Many-Task Computing on Many-Core Computing Platforms”, Doctoral Showcase, IEEE/ACM Supercomputing/SC 2012

[8] Scott Krieder, Ben Grimmer, Ioan Raicu. “Early Experiences in running Many-Task Computing workloads on GPGPUs”, XSEDE 2012

[9] Scott Krieder, Ioan Raicu. "An Overview of Current and Future Computing Accelerator Architectures”, 1st Greater Chicago Area System Research Workshop, 2012

[10] Scott Krieder, Ben Grimmer, Ioan Raicu. “Enabling Many-Task Computing Workloads to Efficiently Harness SIMD Many-Core Architectures”, Technical Report, Illinois Institute of Technology, 2012

[11] Parallel Programming and Computing Platform, CUDA, NVIDIA. http://www.nvidia.com/object/cuda_home_new.html, 2012

[12] White Paper: NVIDIA’s Next Generation CUDA Compute Architecture: Fermi, NVIDIA, 2012

[13] Ian Foster, Yong Zhao, Ioan Raicu, Shiyong Lu. “Cloud Computing and Grid Computing 360-Degree Compared”, IEEE Grid Computing Environments (GCE08) 2008

[14] Glen M. Hocky, Thomas E. Markland, and David R. Reichman. Growing Point-to-Set Length Scale Correlates with Growing Relaxation Times in Model Supercooled Liquids. Phys. Rev. Letters, Vol 108,22, pp. 225506-225510, 2012.

[15] Ian Foster, Carl Kesselman, and Steven Tuecke. "The anatomy of the grid: Enabling scalable virtual organizations." International journal of high performance computing applications 15.3 (2001): 200-222

[16] Rajkumar Buyya. "High Performance Cluster Computing: Architectures and Systems (Volume 1)." Prentice Hall, Upper SaddleRiver, NJ, USA 1 (1999): 999.

[17] Meuer, Hans Werner. "The TOP500 Project. Looking Back over 15 Years of Supercomputing Experience." PIK-Praxis der Informationsverarbeitung und Kommunikation 31.2 (2008): 122-132

[18] Zhao Zhang, Allan Espinosa, Kamil Iskra, Ioan Raicu, Ian Foster, Michael Wilde. "Design and Evaluation of a Collective I/O Model for Loosely-coupled Petascale Programming", IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS08) 2008

[19] Team, IBM Blue Gene. "Overview of the IBM Blue Gene/P project." IBM Journal of Research and Development 52.1/2 (2008): 199-220

[20] Michael Wilde, Mihael Hategan, Justin M. Wozniak, Ben Clifford, Daniel S. Katz, Ian Foster Swift: A language for distributed parallel scripting Parallel Computing 2011

[21] Agarwal, K., Chase, J., Schuchardt, K., Scheibe, T., Palmer, B., Elsethagen, T. Design and Implementation of 'Many Parallel Task' Hybrid Subsurface Model 4th Workshop on Many-Task Computing on Grids and Supercomputers 2011.

[22] Adhikari, A. Peng, J., Wilde, M., Xu, J., Freed, K., Sosnick, T. Modeling large regions in proteins: Applications to loops, termini, and folding Protein Science 2011.

[23] Boker, S., Neale, M., Maes, H., Wilde, M., Spiegel, M., Brick, T., Spies, J., Estabrook, R., Kenny, S., Bates, T., et al. OpenMx: An Open Source Extended Structural Equation Modeling Framework Psychometrika - Vol. 76, No.2, 306-317 April 2011

[24] Woitaszek, M., Dennis, J., Sines, T. Parallel High-resolution Climate Data Analysis using Swift. 4th Workshop on Many-Task Computing on Grids and Supercomputers 2011.

[25] Uram, T., Papka, M., Hereld, M., Wilde, M. A solution looking for lots of problems: Generic Portals for Science Infrastructure Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery 2011.

[26] Wu, W., Uram, T., Wilde, M., Hereld, M., Papka, M. Accelerating Science Gateway Development with Web 2.0 and Swift TG 10 - Proceedings of the 2010 TeraGrid Conference 2010.

[27] Joe DeBartolo, Glen Hocky, Michael Wilde, Jinbo Xu, Karl F. Freed, and Tobin R. Sosnick Protein Structure Prediction Enhanced with Evolutionary Diversity: SPEED Protein Science Journal Jan 2010.

[28] Andriy Fedorov, Benjamin Clifford, Simon K. Warfield, Ron Kikinis, Nikos Chrisochoides Non-Rigid Registration for Image-Guided Neurosurgery on the TeraGrid: A Case Study College of William and Mary Technical Report 2009.

[29] Stef-Praun, T., Clifford, B., Foster, I., Hasson, U., Hategan, M., Small, S., Wilde, M and Zhao,Y. Accelerating Medical Research using the Swift Workflow System Health Grid 2007.

[30] Stef-Praun, T., Madeira, G., Foster, I., and Townsend, R. Accelerating solution of a moral hazard problem with Swift e-Social Science 2007.

[31] Swift/T: Scalable Data Flow Programming for Many-Task Applications. Justin M. Wozniak., Timothy G. Armstrong, Michael Wilde., Daniel S. Katz., Ewing Lusk., Ian T. Foster. Submitted to PPoPP 2013.

[32] Turbine: A distributed-memory dataflow engine for extreme-scale many-task applications, Justin M. Wozniak, Timothy G. Armstrong, Michael Wilde, Ketan Maheshwari, Daniel S. Katz, Ewing L. Lusk, and Ian T. Foster, The 1st International Workshop on Scalable Workflow Enactment Engines and Technologies (SWEET'12), May 20, 2012, Scottsdale, AZ.

[33] ExM: High Level Dataflow Programming for Extreme-Scale Systems, Timothy G. Armstrong, Justin M. Wozniak, Michael Wilde, Ketan Maheshwari, Daniel S. Katz, Matei Ripeanu, Ewing L. Lusk, Ian T. Foster, 4th USENIX Workshop on Hot Topics in Parallelism (HotPar '12), June 7-8, 2012, Berkeley, CA.