RC24049 (W0609-032) September 12, 2006
Computer Science

IBM Research Report

Forms of Collaboration in High Performance Computing: Exploring Implications for Learning

Catalina Danis
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Forms of Collaboration in High Performance Computing: Exploring Implications for Learning

Catalina Danis
Social Computing Group
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598 USA
danis@us.ibm.com

ABSTRACT
Successful collaboration is not only an occasion for the accomplishment of shared goals, but also provides opportunities for individual collaborators to learn from each other. Extended interaction allows for participants to resolve personal and professional differences and thus create a foundation for successful collaboration. This paper contrasts opportunities for learning in short-term and long-term collaboration in the context of scientists working with High Performance Computing (HPC) system experts. It explores how factors conducive to successful collaboration in longer, more tightly organized collaboration might be adapted in more transient collaboration between scientists and HPC consultants.

Categories and Subject Descriptors
H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – collaborative computing, computer-supported cooperative work.

General Terms
Management, Human Factors.

Keywords
Collaboration, teams, consultants, learning.

1. INTRODUCTION
In addition to providing a means for accomplishing shared goals, scientific collaboration provides an opportunity for participants to learn from each other. In a successful collaboration, mutual learning is promoted through extended working histories during which personal and professional differences can be resolved and a trusting working relationship can be established (4, 5). A collaboration of shorter duration provides less opportunity for mutual learning to take place.

Collaboration has long been recognized as critical for solving complex scientific problems (8). By joining forces with experts who have complementary skills, scientists can explore bigger questions than they could do alone. As parallel computation becomes an increasingly important method for accomplishing scientific work, the need for collaboration among scientists and computational experts is growing. However, since parallel machines (often called High Performance Computing Systems, or HPCS) are still relatively rare due to their expense, many scientists gain access to HPCS resources through supercomputing centers[1] where collaboration with computational experts is, typically, of the more transient variety.

This paper explores factors that might be leveraged to promote learning opportunities in short-term collaboration between scientists and computational experts. In particular, it examines how relatively transient interactions between scientists and computationally skilled consultants might be exploited to teach computation to scientists.

2. COLLABORATION BETWEEN SCIENTISTS AND COMPUTATIONAL EXPERTS
Collaboration with experts who have skills complementary to one's own is an important strategy for addressing complex scientific problems. As the scientific knowledge base grows, scientists are finding that it is not feasible for an individual to master all the necessary skills. One of the informants interviewed for this study, a code optimizer who was trained as an electronic atomic physicist, voiced a common lament heard from scientists whose work can benefit from the use of HPC machines: Science and computation place conflicting time demands on the scientist. He described a typical path for a scientist who learns computation: "… you learn Fortran, write a program … Because you don't know a lot about computing, your code does not run well, but you spend little time on making it work better because it interferes with the activities you need to survive as a scientist, like publishing….". In order to make one's code run better, one needs to optimize it for a specific machine. This places additional demands on the scientist because "… you need to develop a body of knowledge about hardware, memory, protocols. Basically, you

[1] Supercomputing Centers in the United States are funded by government scientific agencies to provide access to human and machine HPCS resources to qualified scientists.

need to dig deeper, not in Computer Science, but in Computer Engineering. …I am no longer a practicing scientist."

While the first supercomputers, as HPC machines are known in the vernacular, were delivered in the early 1960's, the use of parallel computation has been, until recently, largely limited to elite government laboratories (2). But, as costs of machines have started to decrease, use is spreading to commercial and scientific areas and parallel computation is increasingly being taught in Computer Science departments at universities (2, 7). However, instruction in parallel computing for scientists during university training is lagging training for computer scientists (1). Consequently, scientists have to learn parallel computation "on-the-job."

For scientists, access to on-the-job training in computation varies. A successful instance has been identified in a recent study of the organization of code development teams at one of the national laboratories in the United States (3). There the scientific coding work is carried out by teams composed of scientists and computational experts. Since many of these are long-lived, there is considerable opportunity for mutual learning to take place. However, many university scientists do not have access to HPC resources – neither human nor machine – at their universities. Thus, the necessity for the supercomputing center resource model.

This paper explores how short-term collaborative interactions might provide opportunities for less computationally skilled scientists to acquire computational skills. It explores the nature of the collaboration that takes place and identifies factors that may obstruct and those that may facilitate collaborative learning. It does this through contrasting two forms of collaborative relationships: long-term, enduring teams and short-term, transient collaboration that occurs through consultation.

3. SITES AND STUDY METHODS

The data in this paper are drawn from interviews with ten individuals who are highly skilled at computation for HPC systems. Five (four consultants and their manager) work at one of the supercomputing centers in the United States that is funded by the National Science Foundation, a government agency. Scientists gain access to the center's resources by writing a proposal requesting computer time. The criteria for selection at this particular supercomputing center includes demonstration that the scientist's application can scale to the terascale capabilities of the center's machines. Once accepted, the scientist can request computational help from the center's consultant staff. All four consultants and their manager have post-graduate degrees in science. In addition to their consulting work, they develop HPC codes in their areas of scientific interest.

The other five interview participants are current or former employees of an international vendor of HPC hardware and software. Three are current or former employees of the research division who develop tools and applications for experimental HPC systems. The other two are technical employees of the company's HPC sales organization. Their job responsibilities include improving the performance of clients' codes.

The data reported here were gathered primarily through semi-structured interviews designed to elicit information about the educational and work training of the interview participant and to understand his or her current work practices in detail. Each individual was interviewed at least once for between 60 and 90 minutes. Six of the interviews were done face-to-face at each individual's workplace; the others were done over the telephone. Nine of the informants was interviewed only by the author, while the tenth interview included a colleague of the author's. The author made transcriptions soon after an interview using audio-taped records or from handwritten notes taken during the interview. Additional briefer follow-ups, targeted to specific questions were conducted through email, telephone or face-to-face with 5 of the interview participants.

4. FORMS OF COLLABORATION IN HPC

The first form, designated *team collaboration*, is represented here by data from a four person core team that has collaborated intermittently for almost a decade on one evolving code project. Discussed below is a portion of their collaboration which occurred over a period of 1.5 years. The second form, designated *short-term consultancy*, is represented by data from short-term, sporadic interactions between a scientist, who is the primary author of a code, and a consultant, who has been assigned to help him or her to bring the code to production readiness on one of the supercomputing center's machines. The discussion of the short-term consultancy includes examples from four consultants.

The two forms of collaboration differ in terms of how intensely the work of the collaborators is coordinated, the duration of the collaboration (years vs. weeks), the role of the computational expert (part of the core team vs. consultant) and the nature of the information exchange (face-to-face vs. email or telephone). In addition, while the participants in both forms share the overall goal of producing production-level code, the consultants do not share the scientists' scientific goals.

4.1 Team Collaboration

To illustrate team collaboration, I draw on examples from a team that in 2003 completed an extensive simulation of earthquake damage to buildings in the San Fernando Valley. The underlying science is about the propagation of waves through materials and the faults that set off the earthquakes. The core team included two civil engineers with expertise in the impact of earth movement on structures built on various types of soils, a computer scientist with expertise in irregular meshes which are needed to map a physical area onto an appropriate data structure and the informant whose role was that of a code optimizer. His expertise is in areas such as communication bandwidth, latency, I/O and he has a deep understanding of the particular machines that were used for production runs of the earthquake codes. In testament to its scientific achievement, the code was awarded the prestigious Gordon Bell prize which recognizes a code of scientific merit that achieves the best raw performance on an HPC machine.

The core members of this team had been collaborating for several years by the time the work discussed here took place, and seem to have followed a style that Hara et al. (4) call integrative collaboration. The core members' distinct areas of expertise allowed them to partition the work so as to work independently during long periods[2]. But, after several months, they would meet for month-long co-located periods of tightly coupled collaboration during which they would test the combined code on

[2] Interviews with other members of the team may have revealed more closely coupled collaboration during these periods.

the production machine. It is during these periods that needs and opportunities for mutual learning and cross-influence arose.

As all of the team members were computationally skilled, the co-located periods were opportunities to combine the various code modules and test the whole on the production machine. Many test runs were done to determine how the algorithms that had been developed over the previous six months would perform on the target machine. Considerable iteration took place in the codes as a result of the performance measurements. To be effective, the code optimizer had to learn enough about the science and the data structures to be able to suggest changes that would exploit the capabilities of the target machine yet preserve the scientific validity of the algorithms. The algorithmic changes that were made by the individual team members in response to data from the test runs were critical for scaling the code to test a problem of large enough magnitude to be of scientific interest – mapping an area 80 kilometers square by 30 kilometers deep of the San Fernando Valley. Without integrating the knowledge held by the code optimizer, only a smaller, less scientifically significant problem would have been solvable.

In addition to the opportunities for learning from each other that repeated co-located work sessions afforded the team members, additional credit was given to one of the co-principal investigators for his skill at assembling a "learning organization." He emphasized the need for all team members, including the array of relatively transient graduate students and post-doctoral fellows, to learn about each other's work.

An appreciation of the scientific accomplishment of one's team members is that it both helped "avoid making unreasonable demands of each other" and conversely, prevented "push-back" when others' demands were reasonable, but costly for oneself. In any collaborative problem solving effort, there are bound to be differences of opinion on how to solve a problem (5). These can escalate, especially if alternative solutions have a disproportionate impact on the work of one sub-group. For example, the code optimizer described an incident in which a particular partitioning of the data across processors assumed by the data structures expert resulted in significant engineering challenges for the optimizer. The data configuration was straining capabilities of network latency and performance of I/O nodes. Given the challenges that this created for the optimizer, he could have resisted the recommendation of the data structures expert. However, since the optimizer could appreciate the goal of his colleague, which was to load balance the system in order to use all available processors efficiently, he worked to figure out a way to make it work.

In these examples, the extended nature of the collaboration both creates the needs and the opportunities for mutual learning. Through learning from each other, the team is better able to achieve its scientific goals.

4.2 Short-term Consultancy

The role of the consultant, at the supercomputing center where interviews were carried out, is to "…help the scientist fix any problems that prevent the code from achieving a production run." The amount of help provided by the consultant depends on the scientist's level of computational skill. The more sophisticated users only need assistance related to the particular HPC machine and infrastructure in use at the center. For example, one scientist required help with scientific debugging the operation of the function MPI[3]_Wait that stemmed from different implementations of the MPI library in use at the center and the scientist's university. Less-skilled users required more extensive help related to programming of the problem, sometimes even requiring help with serial constructs.

Scientists initiate contact with the consulting staff through an email to the supercomputing center's "hotline" after their proposal for use of the center's resources has been approved. The head of the support organization then assigns a Scientific Computation Consultant to work with the scientist, matching domain expertise whenever possible. Once the scientist's code is completely debugged it enters the production run stage and a consultant in the Runtime Support group assumes responsibility for the remaining production runs. As this may take several weeks to months (most codes share the HPC machine at the center with many other users), there is ample opportunity for interaction with a consultant. The focus for the remaining discussion is on collaboration between a scientist and the Scientific Computation Consultant, who works with multiple scientists concurrently.

A domain match between a scientist and a Scientific Consultant is not necessary for the collaboration to be successful. For example, one consultant reported how he resolved a memory leak problem in a scientist's code. He noted that solving many of his clients' problems "… requires careful analysis – we may not know the science behind it but we know what the program is doing".

There were cases, however, when the "common ground" (1) that derives from shared intellectual background is required for the consultant to provide value to the scientist. For example, most Quantum Chemists do not write code from scratch, but instead use one of the several "packages" that are sold in this computationally mature area. The concept of packages in Quantum Chemistry is similar to statistical packages used by social scientists: Standard types of analyses are encoded in pre-specified methods into which the user inserts her data. The consultant needs to understand the models behind the various analyses in order to advise the scientist appropriately.

The same is true in some cases when standard HPC programming languages, rather than domain-specific packages, are used. This is particularly true in cases where the problem occurs in the algorithmic portion of the code designed to carry out the scientific work. The consultant without knowledge of the scientist's domain may not be able to detect problems should the scientist's intent be violated by the computation, even though the program completes without error. In such cases, the consultant depends on the scientist to detect that the results "don't seem right."

The distributed nature of the collaboration, as well as the competing demands on each member's attention introduce inefficiencies into the collaboration and dilute opportunities for mutual learning (6). For example, in one case that involved a computationally knowledgeable scientist, one described as a "good type of user; one who could isolate what he needed," who only needed help adapting his code to the center's infrastructure, it took a pair more than two weeks to determine the cause of invalid scientific results from production runs. Because the scientist was computationally knowledgeable, he was able to

[3] The Message Passing Interface is a library of parallel constructs added to C or Fortran to enable writing of parallel code.

determine that the program "… wasn't working as he had expected," even though it ran without errors. However, the pair failed to resolve the problem with several rounds of emails back and forth. Eventually, the scientist created a short, one page program that isolated the problem and the pair was able to trace the root cause to the different implementations of the programming language at the center and scientist's serial machine (where he wrote and debugged his code prior to submitting it to the center). The implementation differences had masked the scientist's misunderstanding of the method for doing data updates.

Another example of "universal problems that collaborators need to resolve" (5) concerns occasional disagreements on the division of labor between scientist and consultant. One consultant reported a case of a professor who was having many problems with using a Quantum Chemistry package. He developed a work-around for one of the problems that he had traced to a defect in the version of the package in use at the center. However, he felt that the scientist should resolve another problem that had to do with the data structures that she had created. While she was inexperienced in the use of the current package, the consultant concluded that she should be able to leverage her experience with another package to solve the current problem. However, she pushed back, giving him the impression that "she wanted me to do it for her." Another consultant also noted that many users are only interested in getting their code to run, and are unwilling to work to make it run well. He noted that there are "all kinds of users who are capable but it takes quite a time commitment to do that; but most users are not interested in it." This sentiment is reminiscent of the feelings expressed by the atomic physicist turned optimizer quoted early in this paper: Computation can seem like a distraction from the main work of the scientist.

In spite of the above examples of challenges that arise in the short-term, consultancy form of collaboration, there are some factors that might be leveraged to increase opportunities for scientists to learn computation skills.

The Scientific Computation Consultants reported that occasionally the scientists reverted to asking them for help after responsibility for their codes had been passed to the Runtime Support staff. One consultant hypothesized that this might indicate that the scientist had gotten comfortable with him and perhaps felt he could rely on him. The preference thus expressed by these scientists is reinforced by the center's practice of assigning a returning user to the consultant they worked with previously. Whether this is done intentionally to develop working relationships or as a side-effect of matching domain expertise, the practice extends the collaboration though time and may episodically begin to mimic longer-term collaboration.

Another possibility is trying to shape the scientist's behavior during the short-term collaboration episode. For example, one consultant reported that he tried to enforce good programming practice as a condition for helping users. He had grown dismayed that some "users are stuck in the old practices" and fail to take advantage of newer tools that can make them more efficient. He tried to shape better programming practice, by, for example, giving his scientific collaborators an ultimatum around declaring all variables in a single place even though the language they were using in combination with MPI, Fortran does not require it. "I tell users that if you don't use it you are on your own. I keep telling him 'hey you have to put this in; it's good practice."

5. CONCLUSIONS AND FUTURE WORK

These data support the expected differences between the value provided between *team collaborations* and *short-term consultancies*. With more interaction, there is more opportunity to develop shared understandings, more reason to make commitments to accommodate each other (4, 5, 6). Learning from each other is not only necessary but also possible. However, the picture for the more transient collaborations is not hopeless. Some of these consultants demonstrate that they have some leverage with their users to encourage them to learn more, though the generality of this is yet to be explored. Repeated interactions may also provide occasions for more learning opportunities to occur naturally.

This is preliminary work and requires further investigation. The next step is to include the scientists in subsequent interviews, so that a more complete view of the collaborative relationships may be obtained.

6. ACKNOWLEDGEMENTS

My thanks to the consultants and managers who shared their experiences with me.

7. REFERENCES

[1] Clark, H.H. and Brennan, S.E. *Grounding in Communications,* in Perspectives on Socially Shared Cognition, Resnick, L.B., Levine, J.M. and Teasley, S.D. (Eds.). 1991, APA, p. 127-149.

[2] Graham, S. L., Snir, M., and Patterson, C.A. (Eds.) *Getting Up To Speed: The Future of Supercomputing,* 2005, National Academies Press.

[3] Halverson, C. Personal communication, March 2006.

[4] Hara, N., Solomon, P., Kim, S-L., and Sonnenwald, D. H. *An Emerging View of Scientific Collaboration: Scientists' Perspectives on Collaboration and Factors that Impact Collaboration.* Journal of the American Society for Information Science and Technology, 2003, 54 (10): p. 952-965.

[5] Kraut, R., Galegher, J. and Egido, C. *Relationships and Tasks in Scientific Collaborations.* Human-Computer Communication, 1988, 3: p. 31-58.

[6] Olson, G. O. and Olson. J. S. *Distance Matters.* Journal of Human Computer Interaction, 2000, 15 (2-3): p. 139-178.

[7] Pollock, L. and Jochen, M. *Making Parallel Programming Accessible to Inexperienced Programmers through Cooperative Learning.* In ACM Special Interest Group on Computer Science Education (SIGCSE '01), ACM: New York, p. 224-228.

[8] Weinberg, A.M. *Impact of large-scale science on the United States.* Science, 1961, (134): p. 161-164.