Internetics: Technologies, Applications and Academic Field
or
Parallel Computing and Computational Science Do Not Quite Work

Geoffrey C. Fox
Syracuse University
Northeast Parallel Architectures Center
111 College Place
Syracuse, New York 13244-4100
gcf@npac.syr.edu
http://www.npac.syr.edu

Dedicated to Richard Feynman

Ten years ago, we were all sure that parallel computing technology and the interdisciplinary academic field of computational science would be center pieces of both academic and economic growth. We show that this insight was, in principle, correct but was an incomplete vision for large-scale computation implies both increased computer power and increasing numbers of users and applications. Parallel computing undoubtedly works on essentially all problems, but we were unable to produce deployable software systems. Further, few industries could achieve adequate return to justify investment in parallel computers, except in a few areas such as databases. Computational science is the academic field on the interface of computer science with fields such as physics, chemistry, and applied mathematics. This expertise allows you to be very useful and, in principle, is an excellent area of study, but is not a wise field for many students as employers and universities prefer traditional fields.

We show how parallel computing and computational science has evolved into Internetics, which is a vibrant growing and much larger field that surely does work both in principle and in practice. Internetics embodies the technologies and expertise used in building large-scale distributed systems and linking fields like physics not just with parallel computers, but with the Web of complex heterogeneous computers. This is CORBA and Java, and not just MPI and HPF. It is Internetics that is the emerging academic field, and not computational science, and internetics is of growing attraction to students and employers. Using an Internetics base, we will produce much better software environments for parallel systems, but the commercial and academic fields associated with parallelism will not grow in the near future.

We argue that we almost "got it right" and the essential features of the original vision were correct and are part of current broader thrust.

1 Introduction

In our first book on parallel computing, we joyfully used the well-known fairy tale centered on a mirror that could be asked the question,

"Mirror mirror on the wall - which is the most powerful computer of them all?"

We thought, in 1987, that choice was between a microprocessor-based parallel array, and a traditional vector supercomputer. However, the mirror distorted our vision, and what we should have seen was a distributed array, and not just a closely coupled parallel simulation, but a complex metaproblem with multiple concurrent asynchronous components. There should not be one power user using a single large machine, but communities linked by a geographically distributed ensemble that, incidentally, could include one or more large parallel systems.

In our current vision, applications of interest extend from those in science and engineering to the information area; computing with a hint of communications - the original parallel computing thrust - becomes communications with some large-scale computing; compilers become interpreters; Fortran becomes Java, and many changes like these. We term this overall concept as Internetics, which is the field centered on technologies, applications, and services enabled by worldwide computing and communications. This is defined to be interdisciplinary, as both base technologies and applications are included. It includes issues of large scale from all points of view, not just large individual parallel or distributed compute engines or networks, but also one web client talking to one server. This is large scale for a different reason - it is pervasive.

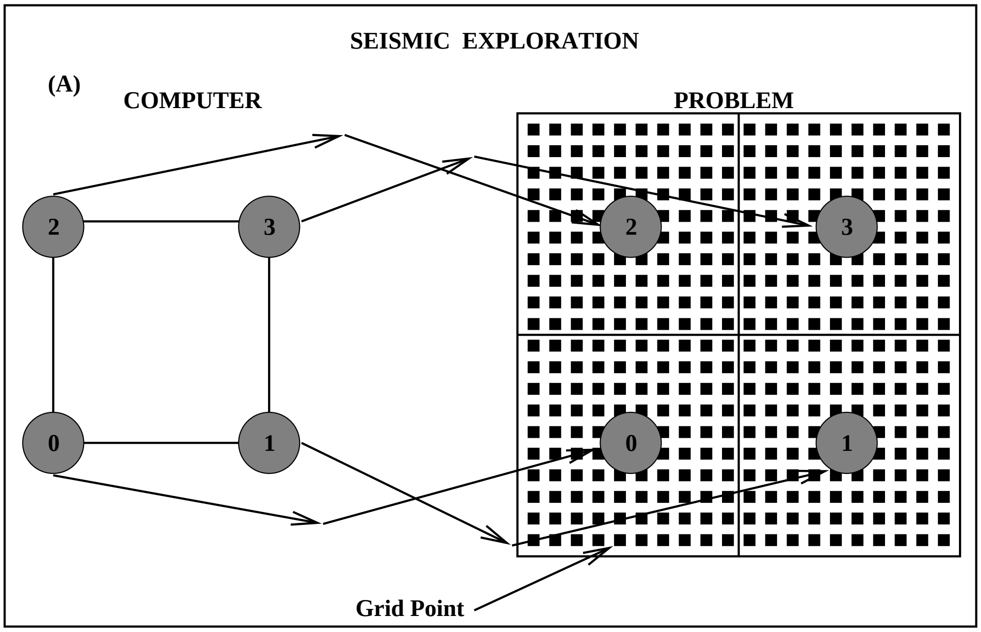
As described in [Fox:91f; Fox:92d], we found in our work at Caltech that interdisciplinary research at the interface of computer science and areas such as physics and chemistry was very rewarding [Fox:94a]. Indeed, individuals who knew both areas, seemed well placed to lead the expected surge of interest in large-scale simulations using parallel systems. Several other groups came to similar conclusions, and the academic field of computational science was set up in several universities, and studied in many conferences [Rice:95e]. However, this initiative seems to have stalled as student interest has shifted in both the computer science and application areas. In the latter case, in fact, fields like physics are seeing a general drop of enrollment as this is perceived as a difficult area to get jobs. Classic computational science (computational physics) is not a large enough field to change this. We argue that internetics combined with physics could, however, offer significant growth opportunities for this and similar traditional fields.

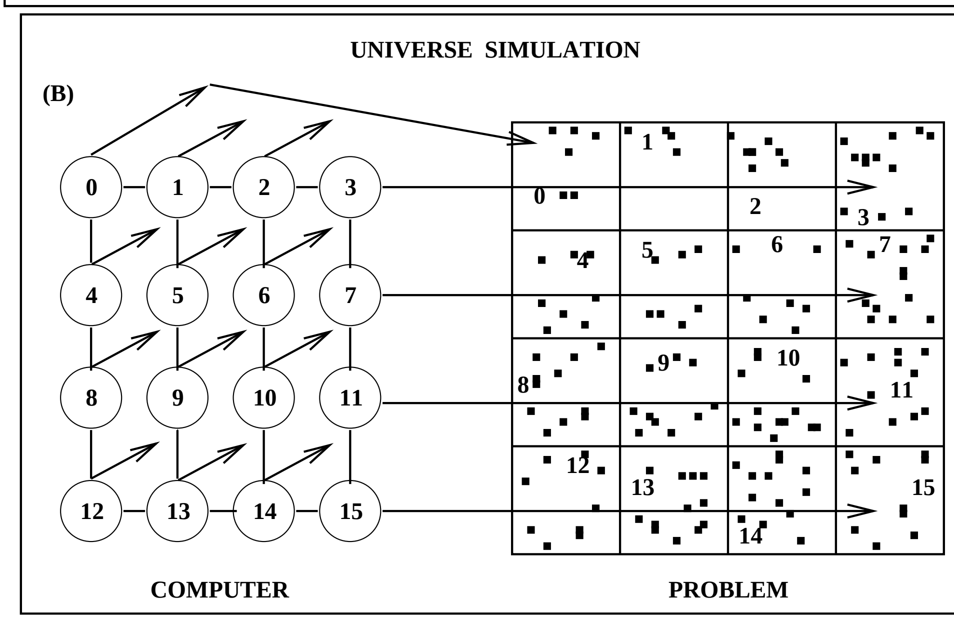
2 Does Parallel Computing Work

It has been clear for some ten years or more than one can parallelize the majority of large-scale applications. Further, this parallelization scales and can be implemented on machines with very many nodes. The essential point is that one only needs to parallelize large problems, and these can be usually thought of as an algorithm applied iteratively to a dataset. The computation is large because the dataset is large. Then parallelism is achieved by breaking the dataset up into parts and placing one part in each processor. In Figures 1 and 2, we illustrate this for the examples we studied at Caltech

- Seismic wave propagation
- Astrophysics
- Computer Chess
- Hadrian's Palace (adapted from an earlier example using his wall)

The datasets are respectively the terrain in which the waves are propagated; the universe in which the galaxies are simulated; the set of moves in a computer generated decision tree; and the set of bricks that the masons must lay to build the wall and tile the floor. The latter analogy (where the "computation" is performed by humans and not digital systems) shows that domain decomposition and parallelism is well known and has established success in all aspects of the human experience. While at Caltech, I use to remark that NASA, when it needed to build a shuttle, did or rather could not hire superman to address the task; rather some 50,000 workers were hired. These built the





shuttles using a dynamic complex heterogeneous decomposition of this single problem. The workers had to be instructed (programmed); be arranged in a hierarchical set of teams (architecture); and the process was designed to ensure workers could proceed effectively and not spend too much time interfering with other team members (minimize inter-processor communication). This was accompanied by dynamic planning and assignment of tasks (adaptive dynamic resource management). The parallel computing terminology is placed in brackets and it is clear that the fundamental computer science issues are familiar concepts in society and that in principal they should be "naturally" soluble. Further, one noted that nature's parallel systems all communicate via message passing whether they be swarms of bees, colonies of ants, collections of neurons, or teams of human minds.

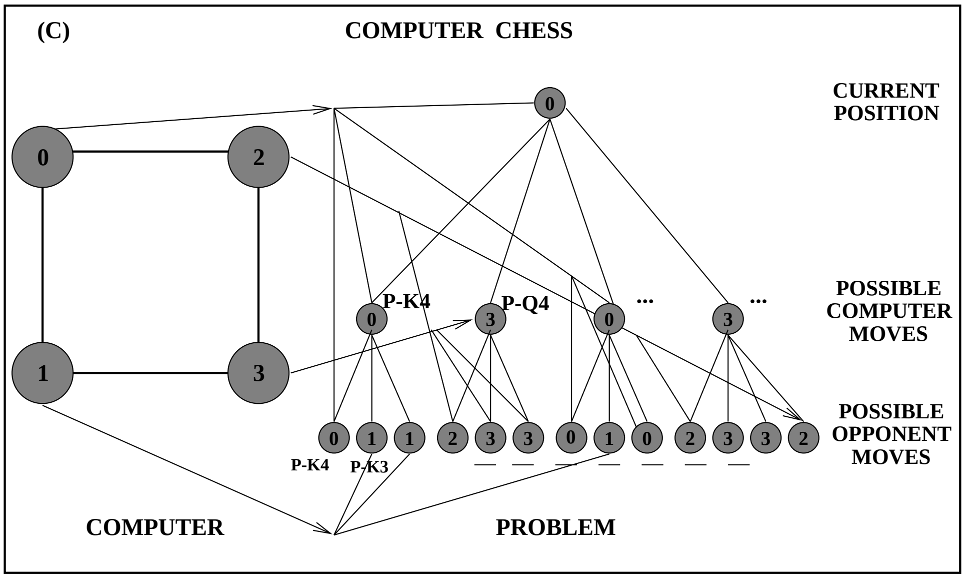
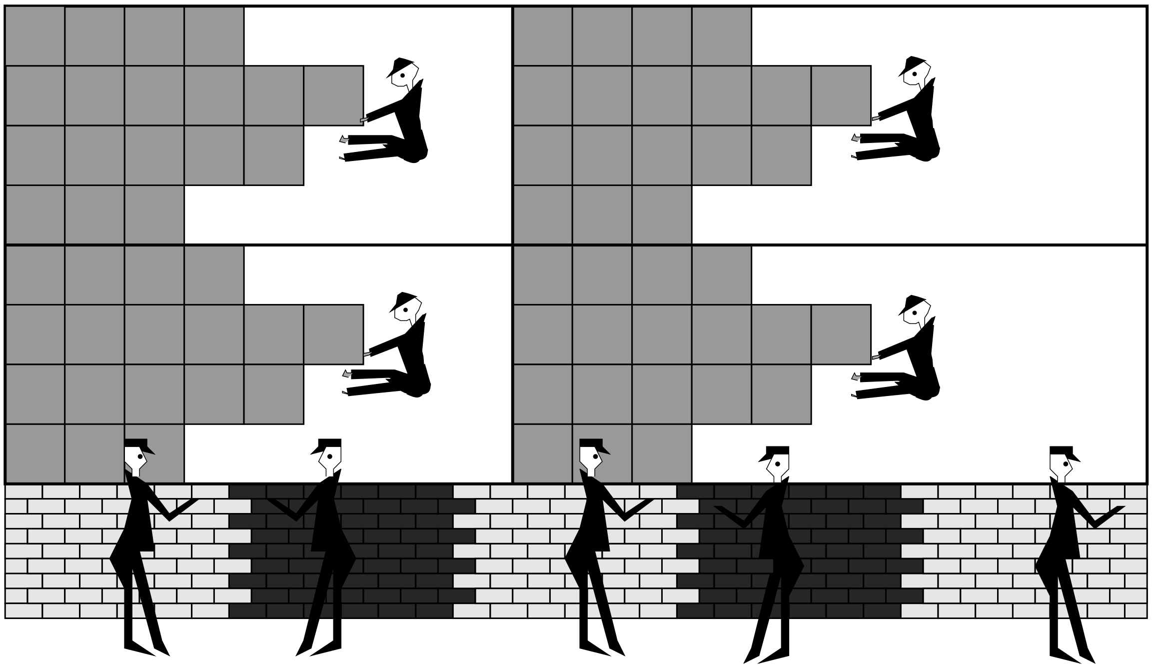


Figure 1: Three examples of parallel computing using domain decomposition to map problem onto computer.

So we were very confident that parallel computing was possible, and set up the "Caltech Concurrent Computation Program" to demonstrate this. The parameters of hypercubes built first by Seitz and then JPL were chosen to support this type of parallelism. As rather tardily demonstrated [Fox:94a], we successfully implemented over 50 distinct significant parallel applications.

Now the above arguments have compelling generality but are, of course, superficial. There are important cases where parallelism is not trivial, including cases where *time* and not *dataset size* is the "large" parameter. Here, we looked at studies of the motion of the solar system, with a few-way parallelism, over long time periods. Solar system studies (using parallelism over planets) cannot use massive parallelism directly but typically planetary evolution is sensitive poorly known initial conditions. These would be studied by multiple runs with different parameter values. This exploratory work is, as they say today, pleasingly parallel (in Feynman's day, it was "embarrassingly" parallel) and recovers our ability to use parallel systems. Gerry Sussman implemented this parallelism with a specialized digital orrery while on sabbatical at Caltech. This, like the hypercube, was discussed in Feynman's class. A second and more important case is event-driven simulations, which are commonly used in the modeling and simulation of macroscopic systems. The military is a major user of this technology and in the U.S.A., this work is coordinated by the DMSO office [dmso]. Event-driven simulations must execute entities in the time ordering of event occurrences and this essentially sequentializes the myriad



Parallel Construction of Hadrians Palace

Figure 2: Tilers laying dance floor in two-dimensional decomposition with masons building the wall with a one-dimensional decomposition.

components. This is *in principle* insoluble, but *in practice* parallelism can be found as events in a large simulation are geographically distributed and do not effect each other for long time periods. This feature, combined with various ingenious variations of the time warp rollback mechanism, actually allows large-scale event-driven simulations to run effectively even on relatively loosely coupled distributed systems. In fact, while at Caltech investigations of such problems was a major activity of our collaboration with the Jet Propulsion Laboratory.

However, these nifty technical issues are not the reason why parallel computing is or is not successful. Rather, the critical point was explained one day in a wonderful public lecture by Carver Mead in Caltech's Beckman Auditorium. He explained how the computing industry faced and would see many technology transitions. However, any new approach needed enough "headroom" to replace the old way. Changes in deployed technology are like the phase transitions in physics to which Feynman made so many contributions.

Systems can live for a long time in a "false minimum" that is the older technology if there is a substantial energy barrier to change, as shown in Figure 3.

We can use a complex systems language as advocated by Feynman's colleague, Gell Mann, and the Santa Fe Institute. A complex system is a set of interconnected entities that, although governed at a low level by standard laws of physics, have interactions that are best described by a "macroscopic coarse graining". As most enterprises involve some sort of optimization, one can usually associate a phenomenological energy function that

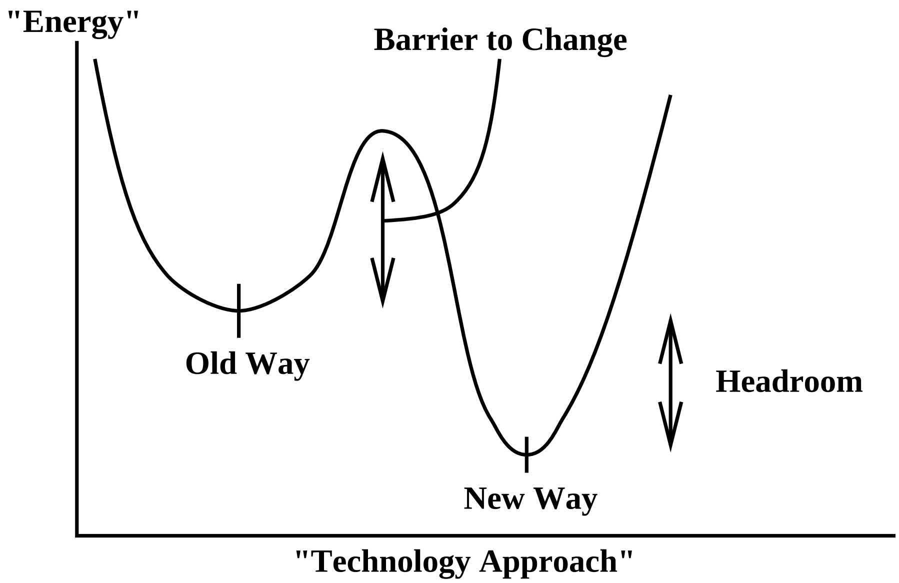


Figure 3: Phase Transitions in Complex Systems

is minimized by our complex system. For instance, the "no-arbitrage opportunity" used in economic modeling implies that one can view the stock market as a complex system. Trading is the heat bath providing a myriad of microscopic interaction that equilibrates this system, and financial instruments are priced by maximizing value. In the computer industry, market forces equilibrate the system while innovation causes the complex system to evolve while always maximizing some unclear energy function representing customer satisfaction per unit dollar.

After this digression, we return to our discussion of parallel computing. In the framework of Figure 3, taking an over simplified view that captures the essence, there are indeed two minima: the "current sequential computing" and the "large-scale parallel systems" minima. The technical computer science studies show that "Parallel Computing Works" so this second minima is distinct, well defined, and lower than the "sequential minimum". However, sequential computing technology has advanced dramatically over the last 15 years and the "headroom" shown in Figure 3 is not so great. We argued that parallel computing was inevitable, as the feature size reduction in chips implied one would "have to spend" one's computer budget on parallel systems as technology reduced the unit sequential system cost. This argument is fallacious for two reasons. Firstly, the industry made the sequential chip architecture "better" as we increased from a few hundred thousand to the current several million transistors in each chip. This did involve parallelism, but only that which could be implemented without user intervention. Current chips are much faster than those of five years ago but, in fact, use transistors less efficiently, as "automatic parallelism" is not as efficient as (user directed) data parallelism. However, this approach stays in the "same minimum" of Figure 3 and requires no "phase transitions". Thus, it is chosen by our market forces. A second, and perhaps more important, development is that effectively users are spending less money each on computing. The dominant thrust in the computing industry is not on a few very powerful systems but on wide spread deployment of very many small (PC's) systems. This is the critical point we missed - large-scale computing was as we always said inevitable, but the scaling included not just the number of processors (as we foretold), but also the number of users. Thus, the dominant system today is not a central closely coupled parallel system linking many individuals and their machine together. There are important problems that still need all the computing they can get. These include large-scale academic computations such as astrophysics and quantum chemistry, and many areas of importance to national security, such as the well-known U.S. Department of Energy ASCI program to model nuclear stockpiles. However, more generally, the anticipated growth has been in this area, but rather in the distributed systems area where the user base has increased at the same rate as the deployment of computer power.

Returning to Figure 3, the "barrier to change" is very large and this is central to understanding why the use of large-scale parallel systems has not expanded. We know good parallel algorithms for almost every important problem and can express these in efficient parallel software. Unfortunately, there are three major difficulties with this process. Firstly, there is usually no easy way to "port (migrate)" existing sequential code to parallel systems. Secondly, the current clean parallel languages are all low level - and, in fact, not much better than what we used in Feynman's courses 10-15 years ago. They are based around explicit user-specified message passing, as in the current PVM and MPI systems today, or the CrOS system we used at Caltech. There are better higher-level systems, but these are not universal "silver bullets" and do not provide a clearly excellent broad-base programming environment, so we have no compelling high-level language that expresses the majority of problems. Even for what we know how to do, there is a further difficulty. Namely, the high-performance parallel-computing field is of order 1% in dollar volume of the commodity computer market. However, it has all the software problems that PCs have plus all the additional parallel computing issues. Thus, the field does not have the capital investment or market size to be able to develop quality software. We have argued recently that this implies that the parallel-computing field should rethink its software strategy [Fox:98a; Fox:98b]. It should build wherever possible on top of software produced for commodity markets, such as the web and business enterprise systems. Here we view parallel computers as a special case of distributed systems with especially tight synchronization constraints.

Note distributed computing assumes problems are already decomposed and designs software to access, store, and integrate decomposed parts together. Parallel computing's central difficulty is different - it is finding a way of expressing tightly integrated problems in a way that they can be efficiently decomposed. We need to focus on this problem, and integrate it with tools taken where possible from the much larger commodity market. Previously, the high-performance parallel-computing community has tried to solve an essentially impossible problem - develop a complete programming environment from scratch with much less available resources than the existing sequential "false minimum" of Figure 3.

Thus, we see that parallel computing established its possibilities but did not, in Carver Mead's terminology, have enough headroom to effect transition from the current sequential "meta-stable equilibrium state".

We can ask if this will change? Firstly, we have oversimplified, as always in such broad-based discussions. There are important areas - especially parallel databases where the transition has been successful. This is understandable because once a single tool (the database) was parallelized, all applications of it could take advantage of parallel machines. Secondly, there is one compelling argument in favor of the inevitable adoption of parallel techniques. One notes that personal computer chips will "soon" have so many transistors that designers must use them to implement parallelism. It is argued that this will force the commodity market to take parallelism seriously, and drive the pervasive deployment of this technology. This argument has some truth to it, but it is not clear that the degree of parallelism involved is enough. If one "just" needs to use up a "few-way" parallelism, then the functional approached (as used by (Java) threads in simultaneously processing different components of a web page) may be sufficient. Such parallelism avoids the critical difficulties of large-scale data parallelism, and will stay in current minimum and not drive the phase transition of Figure 3. Thus, we expect, over the next five years, that the level of activity and importance of parallel computing will remain roughly constant. It will not become the dominant force we thought in the early 1990s. We do expect that parallel-programming environments will improve significantly, but not enough to make it possible to easily leap across the boundary in Figure 3.

3 Is Computational Science a New Academic Field?

So while at Caltech, my studies of both high energy physics and parallel computing were helped immeasurably by an excellent group of students who I diligently trained in an interdisciplinary fashion. I have no doubt that this training was highly effective and was essential for the generally accepted success of our activities. However, these students were typically not so successful on their graduation. Their training was a "jack of all trades" (or at least two trades) and getting a good job - especially in universities - requires excellence in one recognized field. I have, for the last 10 years, recommended students not to perform interdisciplinary work until "they get tenure". This advice flies somewhat in the face of the growing interest in interdisciplinary activities by funding agencies - especially the National Science Foundation. However, not entirely, as one can perform interdisciplinary activities in two ways; firstly, using one or more individuals - each of whose expertise spans multiple fields; secondly, one can build a team of specialized individuals whose combined knowledge spans multiple fields. The latter has, in my opinion, been the mode adopted successfully in most recent projects whereas at Caltech, I largely used the first model. Good universities find it hard to hire interdisciplinary faculty. The tenure review system, in spite of some flaws, has successfully built the quality American research universities. This assumes there is a peer group inside and outside the university that can provide reliable information on which to judge the merits of faculty promotions. This is almost impossible to do in interdisciplinary fields where a candidate, whose work falls into multiple areas, will get less than perfect reviews in any of the component areas of his or her expertise.

There is another more mundane problem with implementing interdisciplinary fields. Suppose we have N basic fields - physics, chemistry, biology, medicine, computer science, electrical engineering, environmental studies, and so on. Then we can design 2^N-1 interdisciplinary areas by choosing any combination of these basic fields. This leads to a plethora of subjects with probably limited life times and no good way to choose where to focus. Thus, it seems best to set up academic institutions with a few core subjects (a basic liberal arts education perhaps?) and building around this an evolving web of interdisciplinary studies. Interdisciplinary work can be recognized by certificates, minors, masters or other "lesser degree" forms. Even this modest goal requires changes in university structure as currently the core subjects have too many requirements and a broader educational experience is hard. However, I believe these changes can and, in fact, probably will be made. In particular, one core field physics is seeing a major reduction in enrollment as it is correctly perceived that there are few jobs in the "pure field" of physics. However, I have found physics is, in fact, an excellent training for general interdisciplinary research. Physics teaches problem solving based on fundamental principles - a good approach in most areas.

My attempts to understand the academic role of computational science revealed another bothering recurring theme. Namely, nobody could agree as to what it was. Individuals at Caltech insisted it was the same as what I call (Σ_i computational *i*) i.e., an amalgam of computational physics, chemistry, etc. However, NSF in its recent "partners for advanced computational infrastructure" solicitation, I think views it, as I do, more broadly as shown in Figure 4. This include as well "applied computer science" or those aspects of computer science involved with hardware, software, and algorithms of scientific and

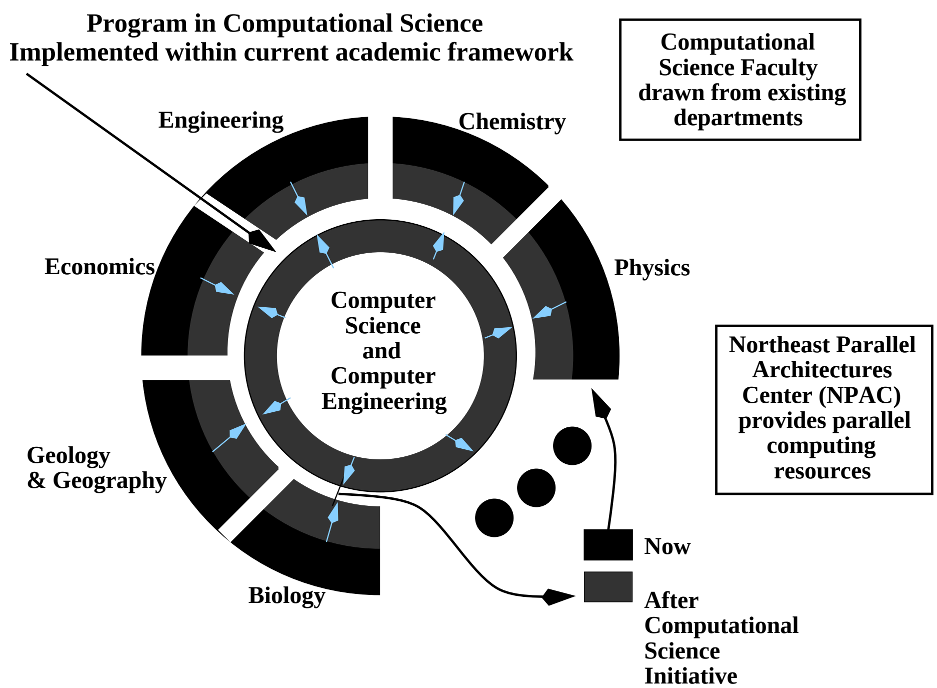


Figure 4: A classical view of computational science as implemented at Syracuse in the early 1990s.

engineering computation. Most academic implementations have, in fact, given computational science a central home in some sort of computer science or applied mathematics department and so emphasizing the last component. The academic computing tower of Babel is further confused by the fields of computer engineering, computational science and engineering, and scientific computing. These take the fields we have discussed and given them different emphases, but leading to, again, good educational opportunities with an unclear national recognition. The major problems for students in computational science is that most employees generally have no idea what the word "computational science" means, and computer science is the best academic degree with which to hunt for jobs. This being said, it is also true that it is the applied and not the theoretical computer science skills that most employers demand. Thus, the strategy explained at the beginning of this section is sound. Get a degree labeled by a basic well understood field such as computer science, but do arrange one's studies to obtain "lesser degrees" (master, certificates) demonstrating proficiency in fields like computational science, which teach good practical skills.

After these general remarks on interdisciplinary research, let us discuss computational science defined as the academic field lying in between computer science (and applied mathematics/computer engineering) and the various fields of science and engineering that use high-performance (parallel) computing. We show in Figure 4, the particular view of computational science we developed at Syracuse University. This was designed in accordance with earlier remarks to be implemented within the existing academic framework and not require a new academic unit. This is consistent with approaches at other universities as described in John Rice's fine summary [Rice:95a].

Our vision for the success of this field is well captured in these words from Daedalus [Fox:92d] in 1992.

> "This essay is constructed around a single premise: the inexorable increase in the performance of computers can open up new vistas in essentially all fields. We need skilled people to explore and exploit these possibilities, however, and our educational system is behind the times. Current curricula at grade schools and colleges will not educate students to exploit the possibilities opened up by parallel computers and the emergence of the computational methodology. Furthermore, the young but relatively traditional field of computer science will only give us a small fraction of the scientists in the computational wave that will lead the revolution. Computer scientists will develop the wonderful machines - a critical enabling technology. However, what we need most are *computational scientists* - individuals trained to *use* computers. High-performance computing is critical to the nation's needs. The Gulf War illustrated this in our military, but the future battles will increasingly be economic. Thus, high-performance computers can assure the industrial competitiveness of the nation, but this can only be true if we educate those who can use parallel computers in new ways for industry."

These were sound arguments, but they embody the flaw exposed in Section 2. Computation will open up new vistas in essentially all fields, but parallel computing *on its own* will not. I moved from permanent summer to winter (Caltech to Syracuse) because I wanted to implement the original computational science vision broadly in a university closely linked to the real world (industry). However, I soon realized the flaw as when I surveyed New York State industry [Fox:92e, Fox:94h, Fox:94i, Fox:95d, Fox:96b, Mills:93a], I found little interest in parallel computing.

In fact, at Syracuse, I developed some reasonable core courses in computational science [CPS615, 714 see http://www.npac.syr.edu/Education] but after peaking with some 50 students and two sections in the early 90s, student interest has waned and the current enrollment is down by an order of magnitude. Syracuse students are pretty bright, but they tend to be pragmatic and go to courses where they think there are jobs to be found. Anticipating the discussion of the next section, over the period 1995-98, enrollment in web technology classes has *grown* by an order of magnitude.

So we do need to change university curricula as computing is impacting every field and this must be reflected in the material students earn. However, I now believe that our original vision of computational science was too limited, and not broad enough to survive the inevitable slings and arrow of uncertain technological progress. We do need some key characteristics - in particular, strong flexible core subject curricula with enough latitude that there is room for students to take interdisciplinary studies. This is a sound lesson we have learnt from the experiments in computational science.

4 Internetics - The Correct Vision?

In last two sections, we described parallel computing as the expected driving technology that would increase the role of computation in all fields and so drive a new interdisciplinary academic field of computation science. This vision was not quite right as it did not anticipate the rapid improvement in sequential architectures. Further, it missed a critical feature of "large-scale"; namely, it was more important to scale the number of people involved than to scale the power of individual computers.

The new technology vision is the wide spread deployment of computational devices and communication links with the essentially identical architecture whether in a central massively parallel server or a distributed set of digital set-top boxes in a suburban community. All are linked commodity processors exchanging messages between themselves. This scenario will not be trapped in a niche market that is 1% of the total but rather will overtake all computer systems. In this world, the operating system seen by the users is WebWindows [Fox:96c] as currently seen in the integration of Microsoft's Internet Explorer with the PC Windows Operating System. According to the basic market principles, web technology will lead to the best available software as it addresses the largest possible market, and so can amortize software development costs over the largest possible volume. Further, the web has a particularly good creative model as its modular distributed software design is designed and built by a loosely knit world wide software team. Note we have had previous pervasive software models such as those of IBM mainframes or of the PC itself. The web is different from these, as it is a *pervasive complete* software model that addresses distributed heterogeneous computing. We can regard any other computing system as a special case of the web. Studying parallel computing will start with its distributed computing base, and as mentioned earlier, add support for tight synchronization. Studying military or electronic commerce applications will add security to the mix, while CORBA is naturally linked to the web to satisfy the need for managed distributed objects.

We can follow Xiaoming Li and call the resultant field *Internetics*. This is the study of technologies enabling and applications enabled by the world wide, large-scale, object web hardware and software infrastructure. As shown in Figure 5, this field includes computing, but also a rich collection of networking and information infrastructure, services, and tools. We had realized the importance of this area from our sad survey at

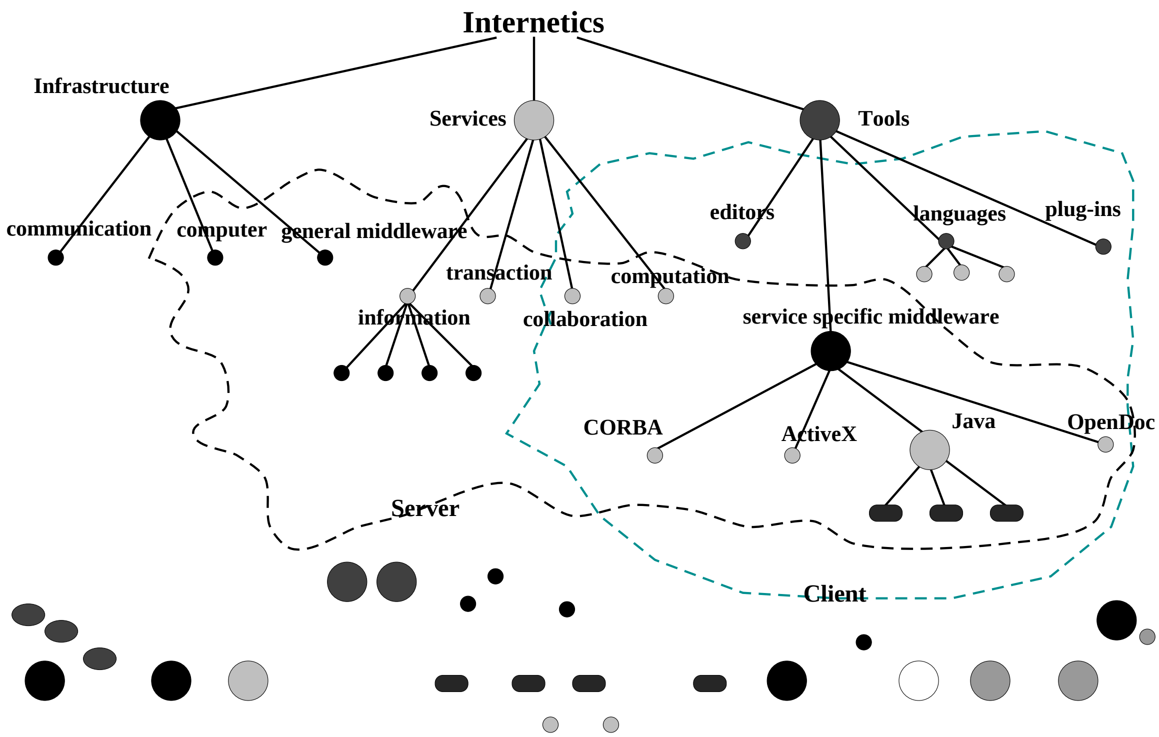


Figure 5: Professor Xiamong Li's view of Internetics

Syracuse, which had shown that industry in New York State was not so interested in parallel computing. We did identify that although large-scale simulation was of "tertiary importance" (as told to me by a now defunct military aircraft company), information processing was of general interest. I also like to recount the tale of a large appliance company that could only find a small, beleaguered audience of six engineers for my talk on the value of simulation and high-performance computing. However, a few years later, they were ecstatic to learn from us how to link their product database to the web.

Even while we were setting up a "classical" computational science program at Syracuse, we realized early on from the feedback from industry that it was incomplete. So, in late 1995, we started an "information track" of the computational science program at Syracuse. The idea is illustrated in Figure 6 and generalizes the concept explained in Section 3 that computational science was at the interface of "applied computer science" and a set of applications. In the information track, we replace the science and engineering applications of Figure 4 with areas shown in Figure 6 where information processing is the key computation task. This includes areas such as education, health care, crisis management, journalism (perhaps using the web for dissemination), and marketing.

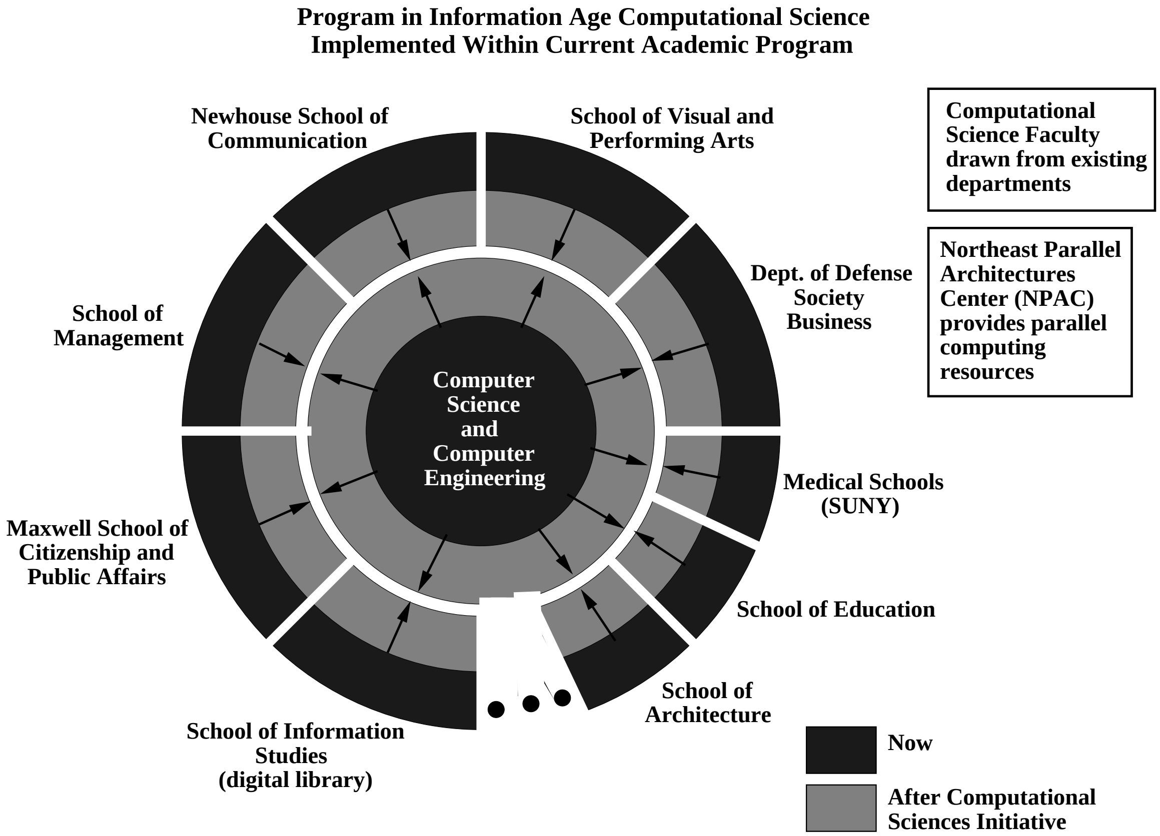


Figure 6: The 1995 extension of Figure 4 at Syracuse to include all applications into computational science. This is a forerunner of an academic implementation of Internetics.

Combining Figure 4 and 6, we find the academic implementation of *Internetics* as the field that lies between modern applied computer science and application areas. We have designed a tentative Internetics curricula running from the K-12 (school children) to graduate level. It starts by teaching school children the essence of the web and how to program in Java. Java is a particularly good language for the K-12 age group, as it has good graphics and obvious utility in improving web pages. Thus, we can easily motivate the language and our beginning programmers get the gratification of better personal web pages to share with their peers. The technologies of Internetics are more social than that of the original computational science. At the graduate level, we designed a six semester course certificate covering technologies (such as the basic Web, VRML, multimedia, collaboration, distributed objects) and a choice of application specializations, such as those in Figures 4 and 6.

We see a general trend towards Internetics (although, of course, typically not with this name) but so far there is not the necessary consensus to expect widespread adoption. For instance, as a subset of Internetics, an interesting field is called by some just "multimedia" and Syracuse scoped this out, but did not adopt a "masters in multimedia" program. However, we see that Internetics embodies the essential vision of computational science that the use of modern computers and communications systems will revolutionize many fields. Thus, it is essential for both academic and economic reasons to train a generation of students to be familiar with both computing and particular applications. In fact, there is substantial interest from industry in retraining existing workers in the techniques of Internetics.

Thus, although details of our original vision were flawed, it is included in the new broader picture, which will succeed.

Acknowledgement

These ideas owe a lot to my interactions with many people. Feynman taught me much about how one should conduct scientific research maintaining both practical relevance and theoretical depth. The members of the Caltech Concurrent Computation Program (C^3P) taught me interdisciplinary research while Professor Xiaoming Li from Peking has been a wonderful collaborator in many ways during the last few years.

References

[dmso] *High Level Architecture and Run-Time Infrastructure* by DoD Modeling and Simulation Office (DMSO), http://www.dmso.mil/hla

[Fox:91f] Fox, G. C., "Achievements and Prospects for Parallel Computing," *Concurrency: Practice and Experience*, 3(6):725-739, December 1991. Special Issue: Practical Parallel Computing: Status and Prospects. Guest Editors: Paul Messina and Almerico Murli. SCCS-29b, C3P-297b, CRPC-TR90083.

[Fox:92d] Fox, G. C. "Parallel Computing and Education," *Daedalus Journal of the American Academy of Arts and Sciences*, 121(1):111-118, 1992. CRPC-TR91123, SCCS-83. Caltech Report C3P-958.

[Fox:92e] Fox, G. C. "Parallel Computing in Industry - An Initial Survey," in *Proceedings of Fifth Australian Supercomputing Conference (supplement)*, pages 1-10. Communications Services, Melbourne, December 1992. Held at World Congress Centre, Melbourne, Australia. Syracuse University, NPAC Technical Report SCCS-302b. CRPC-TR92219.

[Fox:94a] Fox, G. C., Messina, P. C., and Williams R. D., editors. "*Parallel Computing Works!*", Morgan Kaufmann Publishers, San Francisco, CA, 1994. http://www.infomall.org/npac/pcw/.

[Fox:94h] Fox, G., Hawick, K., Podgorny, M., and Mills, K. "The Electronic InfoMall - HPCN Enabling Industry and Commerce," Volume 919 of *Lecture Notes in Computer Science*, pages 360-365. Springer-Verlag, November 1994. Syracuse University, NPAC Technical Report SCCS-665.

[Fox:94i] Fox, G. C. "Involvement of Industry in the National High Performance Computing and Communication Enterprise," in *Developing a Computer Science Agenda for High Performance Computing*, edited by U. Vishkin, ACM Press. Syracuse University, NPAC Technical Report SCCS-716, Syracuse, NY, May 1994..

[Fox:95d] Fox, G. C., and Furmanski, W. "The Use of the National Information Infrastructure and High Performance Computers in Industry," in *Proceedings of the Second International Conference on Massively Parallel Processing using Optical Interconnections*, pages 298-312, Los Alamitos, CA, October 1995. IEEE Computer Society Press. Syracuse University, NPAC Technical Report SCCS-732.

[Fox:96b] Fox, G. C. "An Application Perspective on High-Performance Computing and Communications," *RCI, Ltd*. (to be published). Syracuse University, NPAC Technical Report SCCS-757, April 1996.

[Fox:96c] Fox, G. C., and Furmanski, W. "SNAP, Crackle, WebWindows!," *RCI, Ltd., Management White Paper*, (29), 1996. Syracuse University, NPAC Technical Report SCCS-758.

[Fox:98a] Fox, G. C., and Furmanski, W. "Parallel and Distributed Computing using Pervasive Web and Object Technologies," in Proceedings of ParCo Conference, Bonn, September 16-19, 1997 (to be published). Syracuse University, NPAC Technical Report SCCS-807, February 1998.

[Fox:98b] Fox, G. C., and Furmanski, W. "High Performance Commodity Computing," in Computational Grids: The Future in High Performance Distributed Computing, C. Kesselman and I. Foster, editors, 1998 (to be published). Syracuse University, NPAC Technical Report SCCS-808, February 1998.

[Rice:95a] Rice, J. R., "Computational Science and the Future of Computing Research," in *IEEE Computational Science and Engineering*, 2(4) 35-41, 1995.