were reported for a helix structure and for ions collectively accelerated from laser-produced plasmas; ion energies up to several MeV per nucleon are routinely achieved. New collective accelerator ideas include use of a space charge wave instability in a dielectric guide and a travelling magnetic wave on a toroidal electron cloud.

Laser accelerator schemes, and particle simulations of them, attracted discussion. Some of these schemes employ lasers to produce a beatwave in a plasma at the electron plasma frequency. A very large electric field results over a very small region. It is speculated that very high electron energies (about 1 TeV) might be produced, but very high laser intensities are needed and the number of accelerated particles would be small. No experiments were reported for these concepts.

Several novel ideas were presented that deal with more conventional technologies. A racetrack induction accelerator was proposed with stellerator windings on the curved section to provide beam stability. A beam extracted from an induction accelerator was focused with a series of foils. And a transverse focusing field accelerator was proposed that produces a ribbon-shaped beam that is focused and accelerated between pairs of curved plates with alternating curvatures. An interesting design for a high-current induction accelerator with nine parallel beam channels (Hermes III) was presented.

The closing session offered three talks by renowned experts on areas of special interest. Eric Vogt from TRIUMF reviewed electron and heavy-ion machines for nuclear research. He denoted these approaches as conservative and speculative, respectively, and both have strong adherents. Charlie Baltay of Columbia gave a vigorous report on the Aspen reaction on future high energy facilities in the USA — what technology will be sufficiently powerful to go beyond the Tevatron, LEP, etc? What Laboratory will be big enough to hold the Desertron — the accelerator which will step into, or across, the 'desert' just over the present energy horizon? Finally, Pief Panofsky reminded us that it is difficult to see too far ahead. The utility of a new machine can be different fron our expectations, and plans may therefore need to change. He encouraged the planners: although the cost scaling rules for increasing energy look ominous. Laboratories have reduced unit cost per MeV to the point where total costs are not dominated by the accelerator alone. With development of the necessary talents for these great facilities, we always seem able to make the next step.

(We are grateful to Bob Jameson and 01 in van Dyck for organizing coverage of this Conference and providing the information for this article.)

Theoretical science and the future of large scale computing

Kenneth G. Wilson

There are extraordinary changes taking place in the business community, driven by the twin pressures of Japan and the computer. The change is not always recognized in the academic community or in government, which evolve much more slowly.

The timescale for research and development is shrinking fast. The old picture of research and development can be illustrated by the laser, discovered more than twenty years ago. Now there is going to be a revolution in communications based on lasers and optical fibres. In that twenty-year period, the laser has gone from being a scientific curiosity to the subject of a standard industrial R and D operation. But especially in the computing business, one no longer has twenty years to do R and D. One has maybe three or five years. A product lasts for three to five years, and then it's back to the drawing board. This pressure means that the style of R and D which tinkers with a well-defined object does not work any more. It also requires greater scientific understanding, to enable moving into new areas faster.

In the traditional industrial approach everything inside the industry is secret. Progress is now towards a situation where to gain industrial advantage companies have to learn early on about new developments

The computer message

This article is based on a talk given by Kenneth Wilson at CERN in the CERN Computer Seminar series. Wilson, who was awarded the Nobel Prize for Physics last year (see December 1982 issue, page 403), is a strong advocate of extending and improving the use made of computers in physics.

Over the years, physics and computer science have tended to go their own ways, and although many physicists use computers in their work, their methods are often highly traditional. However, it is becoming clear that a lot of present activity in computer science and software engineering is relevant to high energy physics.

Last year, a workshop on high energy physics software was held at CERN with the subtitle — ' Where do we go from here?' Its aim was to stimulate exchanges of programming experience between computer oriented people in high energy physics and in other professions and fields of research. This meeting provided a glimpse of how physics could benefit from computing developments in other areas. However few lines of direct communication exist, and recommendations were made for ways of improving this awareness. Following what Ken Wilson has to say on the subject is surely one of them.

outside. This will be more important than keeping things secret. Industries have now to reorganize so that outsiders can talk to people in industry and bring ideas in.

This kind of change means that the role of science and scientists in society is going to become quite different from what it has been in the past. Scientific operations should be much more integrated with business operations as business seeks to be informed and to take advantage of basic research. The lead area where this is going to take place is computing because it is here where the R and D times are the shortest.

How will computing developments affect basic research? In elementary particle physics, experimentalists have to analyse events which include hundreds of particles, and a Monte Carlo simulation clearly eats up a major amount of computing power. But the principle of energy conservation helps. Hundreds of particles are involved, rather than millions. But today's theorists face the problem of having to study on a very short timescale the events that the experimentalists measure. Hundreds of particles are not suited to simple analytic theories like the theory of the hydrogen atom or the theory of the earth going around the sun. One has to rely, at least in part, on computer simulation.

Computer simulation for the theorist has to deal not with the final outcome of an experiment, so much as with the factors that govern its final outcome. In a very short time interval energy conservation no longer restricts the number of particles. To be precise, when dealing with short-lived gluons inside a proton or neutron, ten or fifty are not enough. In fact a lattice is needed in order to have a finite number of gluons. But even a finite lattice inside the proton is going to involve thousands, if not millions of points, and gluons should be present at every point of that mesh. The computing power needed for these theoretical simulations; which are still Monte Carlo simulations, are enormously larger than experimental requirements, simply because the number of objects involved is so much larger. And that is why I started thinking about how one would justify getting enormous amounts of computing power into theoretical science.

When I added up the total computing requirements that I needed, I discovered nobody was making that kind of computer. So the first problem was not that I did not have any money, but that even if I did, there would be nothing to spend it on. And pursuing that same line of reasoning, I have gradually come to get an overview of the computing situation, not only in universities but in industry as well, where the big need lies.

One problem is the question of training. The standard attitude of physicists towards computing is that to be ready to do serious computing, whether it be computer simulation for theory or data analysis for experiment, they need only a two-week course in FORTRAN.

'Nobody would suggest that someone is ready to do serious experimental work in physics if they have just had a two-week course in soldering. And yet the attitude is that a two-week course in FORTRAN gets people ready to do computing!'

Spreading the computer message Kenneth Wilson.

(Photo Cornell University)

Now computer simulation for a theorist is extremely close to the idea of an experiment. A computer simulation has to come up with a number as an answer, it does not come up with functions or ideas. So first one has to know how to design a computer simulation whose number is worth getting, which is analogous to setting up an experiment and determining all its parameters. One has to analyse the numbers that come out, and face all the same problems as an analysis of experimental data.

Nobody would suggest that someone is ready to do serious experimental work in physics if they have had just a two-week course in soldering. And yet the attitude is that a two-week course in FORTRAN gets people ready to do computing!

What kind of training is required? Training has two parts. One part is like experimental training — an apprenticeship of several years writing a PhD, working with an experienced person who knows how to formulate an experiment, how to do data analysis, how to make sure that the systematic errors do not wipe out the result, etc. The same set of problems applies to computer simulation. So it is clear that people who want to do that should go through the same kind of apprenticeship.

The other requirement is training in computer science, in the management of large scale software. This has been pooh-poohed by the scientific community. Meanwhile, the scientific community, along with the business community and the national Laboratory community, is building up a backlog of huge piles of FORTRAN which cannot be read or modified and is just an incredible drag on the whole scientific operation. It eats up infinite amounts of people's time which could be better spent elsewhere and directs projects along rigid directions when one really wants to be doing something else!

Throughout the 1960s, industry and US national Laboratories were" busily building these huge programs, not worrying much about what was happening. About 1971 they reached a limit. In US national Laboratories today there are few programs that were written less than ten years ago.

The main thrust in computer science is how to deal with this problem. How do you write large pieces of software so that you know what you have got when you have finished and you can work with it afterwards? Rather than exhortations to write in specific programming languages, computer scientists discuss techniques for software management which apply in any language. Then there is algorithm design and a whole list of topics which have been developed over 20 years.

Any graduate student who is to use a computer as part of his PhD should have about a three-month training course in relevant aspects of computer science. Computer training should also be more integrated with college level courses at both the undergraduate and graduate stage. But this means changing the lan- **Jkjage** or the framework of computing because a student cannot prepare a FORTRAN program in the time that it takes to do a normal course assignment.

Then there is networking. Many physicists face the issue of wanting to work from a home institution on an experiment elsewhere, wanting to talk to colleagues at other institutions, and to exchange data and programs, etc.

There is one additional aspect of networking which I find many scientists are unaware of. This is interdisciplinary communication. For example, an electrical engineer at Cornell was preparing a computer architecture course. He had to cover about *moht* different computers, about half *mf* which he knew, but he needed information on the other half. So he put a request on the ARPANET bulletin board for information about the unfamiliar computers. Experts on those four computers sent him replies over ARPANET.

As more theorists move towards computers, they are moving towards solving problems from first principles. Not just in elementary particle physics where one expects to work from first principles anyway, but in solid state physics, applied physics, chemistry, geology, and many areas of engineering. These people need to communicate with each other so they do not keep reinventing the same thing. The computer network is needed to enable this communication to take place.

This is especially important to bring the academic community in contact with industry as industrial R and D moves from a closed operation which would not accept information, even if it were supplied, to an open operation seeking information. Industrial research people can use the network to ask for help from outside. I am pressing very hard in Washington for a computer network serving all scientists.

Next there is software. The software problem led in the end to the stop in development both at big Laboratories and in business. I think this was a primary reason the demand did not appear immediately for a supercomputer like the Cray. If the demand had developed in the normal way from progress in the 60s, supercomputers would be commonplace now.

The software problem is one word - FORTRAN. FORTRAN is restrictive because there is a limit to the complexity of the programs that one can write using it. FORTRAN has two problems. One is that you cannot read it and the other is that you cannot modify it. Computer scientists have been complaining about this for years, but they have not been very helpful. First they touted ALGOL. Then PL/1. Then PASCAL. But as far as I can see none of these languages really solve the problems of FORTRAN.

An analogy shows the heart of the problem. A 60-page FORTRAN program has roughly as much information as an advanced textbook. To take that textbook and give it the quality of FORTRAN program, scramble all the words. For example, trying to figure out a big FORTRAN program requires continually leaping back and forth through the entire listing. Can something be done about this? I have been trying to understand what computer science ideas might be truly helpful and I am convinced that there are ideas, which if properly packaged, would easily supersede FORTRAN. But the ideas have not been packaged for scientific processing. They are being used in other areas like database management or operating systems because the scientific programmers have said that they do not want anything other than FORTRAN, and the computer scientists have taken them at their word. Furthermore the packaging requires somebody who really understands what a scientific programmer wants. This the computer scientists do not understand.

But I would like to give an idea of what programming might be like using some of these computer science ideas in a scientific programming system. The problem is not to eliminate FORTRAN but to demote it to the level of assembly language, a language produced by machines and not by people. It doesn't matter if assembly language is unreadable because you never read it anyway. Or you shouldn't!

Computer scientists are thinking about how to take models which humans have developed over thousands of years to handle complex organizational problems and use the computer in the framework of these models. As an example, take a textbook, a framework of information organized so that it can be read. Sup-

'The scientific community... is building up a backlog of huge piles of FORTRAN which cannot be read or modified and is just an incredible drag on the whole scientific operation/

pose we want a program to look like a textbook. Chapter 1 will list the equations to be solved. (I am talking as a theorist so I am interested in equations!) Chapter 2 introduces the numerical methods to solve these equations. Chapter 3 lists the boundary conditions. Chapter 4 describes the data structures. Later the book deals with the optimization methods to make the program superfast on one's favourite computer. In a normal FORTRAN program part of each of these different chapters is present in every loop.

How could this sytem of separate chapters actually work? One way is at the end of Chapter 2, after introducing specific codes for numerical methods, a transformation is defined which inserts that code into the equations in Chapter 1. Such a transformation would say, for example, that an integral sign is to be replaced by a certain piece of code. Transformational systems already exist which come quite close to being able to do that. The advantage of transformations is that the equations in Chapter 1 are part of the final program and therefore if the program is changed, but not the equations, the program does not run. The equations have to conform with the program. If the equations were written in a nice orderly fashion but are not part of the program, then they are the equations for what the program did before, not what it does now.

Scientific input is required for a system which serves scientists, so I have set up the GIBBS project at Cornell of which I am the director but all the workers will be computer scientists. I hope there will be other such projects because it is clear that any single project has a very high probability of failure, as in the case of operating systems.

Finally, there is the hardware question. How are we going to get sufficient computing power to tackle basic theoretical questions, and no doubt experimental questions as well? The main trend in computer components is not that they get faster but that they get cheaper. To get maximum computing power the question is not one of having a single computer which gets faster and faster, but having lots and lots of computers running in parallel.

The industry is becoming increasingly aware that it has to think about parallel processing at all levels. But there is also an incredible opportunity for scientists to get into the development of parallel processing by being guinea-pigs for systems very early in their development, where it takes a PhD physicist to make it work. If industry sees all the things that we have trouble with, then they can go back and fix them. What industry wants is to produce something which a PhD physicist can use in his sleep. Maybe then the man in the street can start to use it.

To illustrate various kinds of parallel processing I shall use a human analogy again, namely an airline counter. The first possibility, of not much interest, is to have one clerk behind the counter. If that clerk cannot keep up with the work, then a development project is launched for a single clerk with four arms!

The next approach to parallel processing is the so-called vector supercomputer architecture. Here there are a number of clerks behind the counter and a queue of customers. The first customer advances to the first clerk, the first clerk pulls out a ticket and the customer with his ticket goes to the second clerk. The second clerk writes the customer's name on the ticket. The customer then goes to the third clerk who puts the destination on the ticket. It is an assembly line.

This form of organization works fine as long as all the customers warn tickets, but what if a customer wants his lost baggage traced? When that happens, the entire assembly line grinds to a halt while one clerk sits there processing lost baggage and only when he is finished does the assembly line start processing tickets again.

Some supercomputers have th same problem. They are incredibly fast in doing multiplies and adds but a complicated address calculation has to be done elsewhere, using a single much slower system.

The next form of organization was illustrated originally by the Illiac 4 and now has been reincarnated in several commercial machines. In this organization the customers line up in front of all the clerks and there is another clerk with a megaphone. He barks out 'Customers advance to counter', and they all come up. The clerk with the megaphone says 'Pull out a ticket' and then proceeds to shout out al the instructions for filling out tickets Again it is very good for producing tickets, just as effective as the ar sembly line approach. But it has e>? actly the same problem. When c single customer wants lost baggage traced, one clerk traces baggage ir response to the instructions from the clerk with the megaphone.

In the actual computers, the pro cessors are connected in a squan array and one processor can onl\ pass information to its nearest neigh bour. That is the most common anc

'What industry wants is something which a PhD physicist can use in his sleep. Maybe then the man in the street can start to use it/

fastest form of communication. For problems which are very regular in structure, like Monte Carlo calculations, in lattice gauge theory, this organization is extremely effective. But for things like tracking surfaces of singularities in a fluid dynamics flow, then the computing requirements at one point are very different from the computing requirements elsewhere.

None of these organizations seem to be very useful for detailed experimental data analysis. Another and more promising form of parallel processing has now been developed, but its only commercial realization to date is expensive and not very cost effective. It uses the same organization used in the most effective airline counters. There is a queue of customers and the customers go to the first available clerk. Each clerk can process any request independently of what any other clerk does. One of the big differences with this approach is that whole subroutines or loops can be handled at a time, so yeach processor can process on its lbwn for a long time before it has to communicate with the rest of the system. This is extremely important because while computer scientists have been studying operating systems for parallel processing for some time, invariably they wind up with an operating system which can handle parallel processing except that every time a message has to go from one process to another it takes about a hundred times longer than they originally desired.

What is needed in parallel processing is a way of cutting down the number of times that any processor has to communicate with another, so huge jobs can run before they have to talk to each other.

Now I would like to finish with a variant on this design which was developed^ New York University. It does not yet exist as a piece of hardware. Only the basic concept has been simulated. It deals with the situation where there are 4000 clerks behind the counter. Then the single queue of customers can be a bottleneck. People cannot get to the clerks fast enough. The new approach is that when one of these clerks finishes with a customer, the clerk heads for the queue. If two clerks collide on the way to the queue, one clerk stands aside and the other goes to the queue and fetches two customers, gives one to his friend and takes the other one back.

With 4000 clerks, each clerk that actually makes it to the customer queue is typically taking 32 or 64 customers and distributing them to the colleagues he bumped into.

The 'ultracomputer' design has 4000 processors and 4000 memory modules and a network of criss-crossing wires and network nodes that enables every processor to access any memory module. But when two processors make a request for the same data location, those requests will collide at a network node and get coalesced into a single request. There are no delays even if all 4000 processors want access to the same memory location, for example when all the processors are trying to work on the same loop. Rather than merely accessing the loop index, they want to increase it by one as they process the loop body. For this purpose the ultracomputer people invented an operation called Fetch and Add. When a request to fetch the value of the index and increase it by one is coalesced, it becomes a request to fetch the index and update it by two. When the result comes back, the value of the index goes back to one processor and the value of the index increased by one goes to the other side, so that each processor gets a different value of the index.

I feel this ultracomputer design is the best for the long range future of parallel processing for scientific work. Artificial intelligence, speech processing and areas like that are moving in a different direction. They are thinking of parallel processing in the sense of tree structures. And they like this organization because it is two dimensional and fits very nicely on a chip. There is a kind of organization which is nice for a number of artificial intelligence problems but is not very useful for physics because there would be a bottleneck with the communications between all the processors.

I have no doubt that in the years ahead we are going to see more of this kind of hardware and probably all the processing frameworks I have described.