Defence Science Journal, Vol 46, No. 4, October 1996, pp 257-267

Parallel Computers–A Personal Overview

G. Venkataraman
Sri Sathya Sai Institute of Higher Learning, Prasanthinilayam-515 134

ABSTRACT

This paper offers a brief and general introduction to parallel computers and a description of the PACE (Processor for Aerodynamic Computation & Evaluation) parallel computer project executed by the Advanced Numerical Research & Analysis Group (ANURAG), Hyderabad. Also included are a brief review of the global scene as well as the author's personal reflections on recent trends in the country in the area of parallel computers.

1. INTRODUCTION

This paper reflects the development of parallel computers in India, and also my own modest involvement with it. Necessarily, this article would have a strong autobiographical slant but that I believe does not require any apology.

Although the basic concept of the digital computer existed even prior to World War II, the machine as such came into existence only as the result of wartime computing pressures, originating partly in the atom bomb project and partly in certain complex ballistic problems.

Soon after the War, many universities abroad supported projects to build digital computers and interestingly, Indian scientists, quite undaunted by the enormous technical problems involved (not to mention the shortage of research funds), made nevertheless, modest efforts to design and build digital computers. I believe that the Jadhavpur University was one of the first to build a computer in the country. A significant effort was also launched in the Tata Institute of Fundamental Research (TIFR) in the early fifties and thus it was that the TIFRAC came into existence. It was quite impressive in size, looked very complex, consumed a lot of power (based, as it was, on vacuum tubes) and boasted of a computational capability which then was substantial. TIFRAC introduced digital computing to a whole generation of young scientists. By today's standards, everything was unbelievably primitive–programming had to be done in assembly language, the input was by paper tape, the storage was in magnetic cores, and the output via, an electric typewriter and yet the TIFRAC was a great turning point. People came to Bombay from all parts of India to use the TIFRAC – such was its reputation; however, for all its size and power consumption, its capability probably did not exceed that of an Intel 8086 (!) which is an eloquent commentary on the phenomenal progress witnessed in the computer field.

The next major landmark on the Indian scene was the acquisition by TIFR of a CDC-3600 mainframe machine somewhere around 1963-64. This was a quantum jump, and all of a sudden Indian scientists were catapulted to the international level in terms of computing, power. Overnight, Fortran became popular, and courses on it were offered in all the major institutes, particularly the Indian Institute of Technologies.

Despite the vigorous and early start, things soon began to slip and while the world marched on we began to stagnate. Barring isolated exceptions, very few large machines were acquired in the country and the few that were available were heavily overloaded. Thus, while more and more people became acquainted with programming, hardly any large programs came to be

Received 15 May 1996

written. Factors which promoted this undesirable situation include: acute lack of foreign exchange, pressure to buy ECIL's TDC machines (perhaps partly justified), the reluctance of the US to sell large mainframes especially after the Pokhran nuclear explosion in 1974, and the poor computer literacy of those who wielded power and made all the decisions about the purchase of computers. Altogether there was a severe computer drought, the magnitude and the implications of which were hardly noticed. It is my personal view that this drought produced a tremendous inertia in creative programming from which we have yet to recover.

2. BACKGROUND TO PARALLEL COMPUTERS

In the late seventies and the eighties, American embargo on the sale of high performance computers to India was a major factor in sustaining, the drought mentioned earlier. Many of our development programmes needed such computers but none could be bought for either love or money. The top of the line machine was the celebrated CRAY but its technology was far too forbidding, for us to imitate. The country was really in a fix when a new ray of hope appeared and that was the parallel computer.

Computers originally had only one CPU. As the power of the CPU increased, so did the capabilities, throughput etc., of the machine as a whole. In a nutshell, the power of the computer flowed from that of the CPU which, in some cases like the CRAY, was quite awesome. The parallel computer concept changed all that, most dramatically in fact.

The idea of parallel computing appears to have occurred to a meteorologist named Robinson long before digital computers were even known. Obviously, Robinson was quite ahead of his time; perhaps in his day he suffered ridicule; later he was forgotten, well almost.

Robinson's idea (translated to the modern context) is simplicity itself. Suppose there is a computational task that is massive. Instead of feeding the entire problem to one mighty CPU (as one would do in the CRAY), why not break up the problem into a whole lot *small tasks* which can be executed in *parallel*? For this purpose one must necessarily employ a battery of CPUs but these need not be giants in the CRAY class, since each CPU would be getting only a small piece of the total action. And since the CPUs all work in parallel, time also is saved; in other words, not only does the job get done but quite speedily as well (depending of course on the speed of the CPUs employed). To complete the description, the outputs of the various CPUs are then collected, collated etc., and the final result delivered to the user. This in brief is how parallel computing can handle super jobs.

3. GENESIS OF THE PACE PROGRAMME

We now come to the PACE parallel computer designed and developed in DRDO. The story begins in March 1987 when I made a brief stopover in Delhi, on my way to Russia. Having some free time at my disposal, I made a call on my good friend Dr VS Arunachalam, then the Scientific Adviser to Raksha Mantri. I must mention that I was at that time in the Department of Atomic Energy (to which Dr Arunachalam also once belonged). In the course of our conversation, Dr Arunachalam drew my attention to developments in the area of parallel computers and said: "We need a super computer for our LCA (Light Combat Aircraft) programme. As you know the CRAY is not available to us. The only alternative is for us to design and build our own super computer, and here the parallel computer approach seems an attractive option. Why don't you come over from Atomic Energy to DRDO and start a parallel computer programme?" I replied that I would think it over.

During my brief stay in Russia, I was all the time thinking of parallel computers. On my return to the Indira Gandhi Centre for Atomic Research (my base then), I had numerous discussions with my colleague Dr Neelakantan on the subject of parallel computers. We came to the conclusion that (a) the parallel computer was indeed a viable alternative to the CRAY, (b) the parallel computer was build able (unlike a CRAY-type machine), and (c) the task was both exciting and challenging. We immediately go back to Dr Arunachalam, and the net result was that by December 1987, both Dr Neelakantan and myself moved over to Hyderabad to join DRDO, start a new laboratory (now known as the Advanced Numerical Research & Analysis Group (ANURAG) and get down to the business of building a parallel computer (later named PACE). Needless to say that since in the

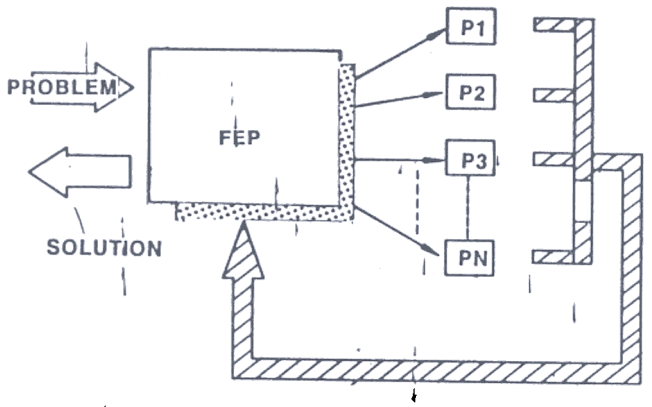

Figure 1. This shows a schematic of a parallel computer. Essentially, it consists of a cluster of CPUs arranged in a suitable architecture. The job is submitted to the front-end processor (FEP) which breaks up the task, distributes the work packages to the various CPUs, collects the results, collates and delivers the output to the user.

beginning there were only the two of us, we literally had to start from a scratch.

4. BASIC STRUCTURE OF A PARALLEL COMPUTER

Figure 1 shows a rough schematic of a parallel computer. The user interacts with a front end processor (FEP) which in some respects is like a reception desk. What it means is that the job is submitted to the FEP. Thereafter. the FEP wears a new hat and plays a different role– that of a Manager. It parcels the job into little packages which are served out to the waiting battery of CPUs. Once the CPUs have completed the number crunching, the FEP collects the individual results, tidies and parcels them up and hands over the final result to the user.

Parallel computing remained essentially a concept till the advent of the microprocessor – that is when the technology took off. This happened roughly around the early eighties and many laboratories abroad tried out the general idea and experimented with various architectures. Among the pioneers was the California Institute of Technology (CALTECH). At CALTECH they configured a parallel computer with 64 Intel 8086 microprocessors. Named the Hypercube (on account of its architecture), the system worked beautifully. True, it was nowhere in the CRAY class as far as speed was concerned but the Hypercube experiment clearly showed that the parallel computer was an eminently

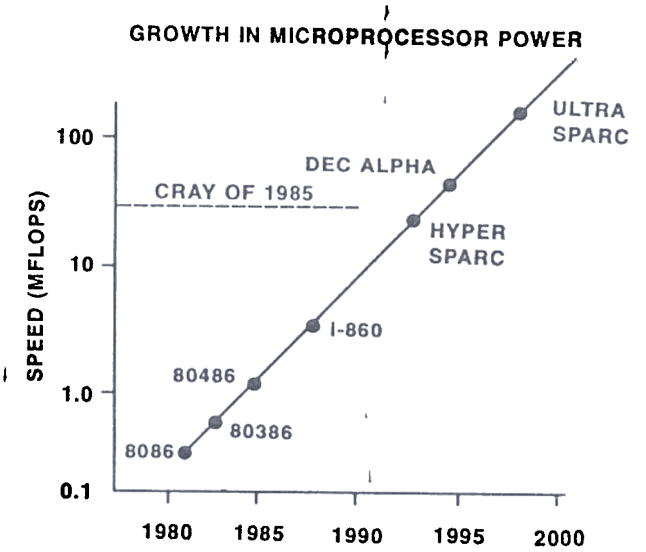

Figure 2. This shows the growth in microprocessor capability over the years. Not all the chips available in the market have been shown.

workable idea. What limited the overall speed was the speed of the individual microprocessor. If that could be improved, then the speed of the parallel computer would also improve likewise. Indeed that precisely is what has happened in recent years. As Fig. 2 shows the power of microprocessors have increased rapidly, far beyond one's wildest dreams so much so that even the CRAY company has now switched over to making parallel computers! Truly, this tale is like that of David and Goliath.

In proclaiming the merits of the parallel computer, I do not wish to convey the impression that they are the panacea for all computational problems. Not so, they are useful only if the computational problem is inherently parallelisable or at least substantially so. Fortunately, many problems in science and engineering are and this is what makes the parallel computer attractive to the technical community. Indeed, experimenting with the CALTECH Hypercube, the scientists of CALTECH demonstrated a satisfying speed up in a wide variety of areas, ranging from astrophysics and cosmology, through aerodynamics to materials science and structural engineering.

Figure 3 illustrates a few types of parallelisms commonly encountered. Of these, geometric parallelism is particularly important in the problems of fluid dynamics and engineering structures.

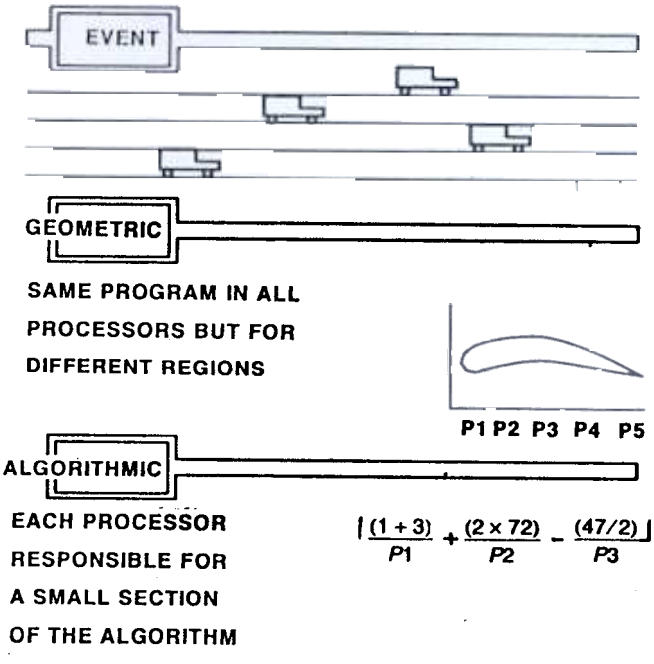

Figure 3. This illustrates some of the different types of parallelism possible. Event parallelism dominates Monte Carlo type problems, while geometric parallelism is at the heart of fluid dynamics.

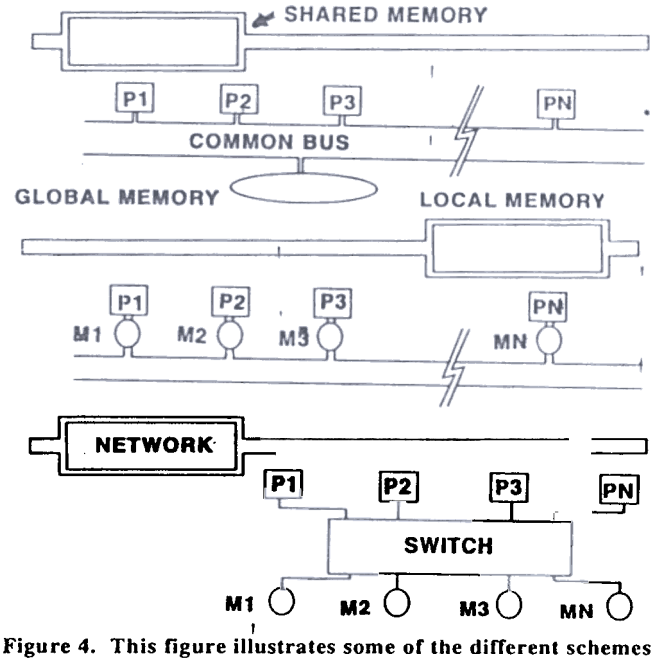

Figure 4. This figure illustrates some of the different schemes of memory management. Explanations are provided in the text.

Turning next to computer memory and its management, Fig. 4 shows a simplified schematic of some of the strategies employed. In (a) is the shared-memory concept is illustrated while in (b), each CPU has its own memory. Yet another possibility is to have a bank of unattached memories and link them to the appropriate CPU via a switching network. In our work in ANURAG, we followed scheme (b).

Next comes the question of architecture or the manner in which the nodes (another name for the CPUs) are linked together. Architecture is particularly important in the context of what is known as message passing. Now although each node has its work cut out by the FEP, the need sometimes arises for the nodes to exchange messages/data between each other and this naturally calls for a suitable communication path. In a system with N nodes, there are, in principle, $N \times (N\text{-}1)$ dialogues possible. If direct links are to be provided to each node, then horrendous complications would arise when N is something like say 1024; other strategies have therefore been explored! In the Hypercube, the nodes represent the corners of a cube in an appropriate dimension, r say. For a r-dimensional Hypercube, $N = 2r$ messages propagate by hopping from node to node, and the architecture would decide through how many nodes must the message hop before reaching its destination. In the Hypercube, the maximum number of hops required is N.

5. BACK TO THE PACE PROGRAMME

Let me now resume the story of ANURAG. Our first job was to assemble a suitable team. Even as we were endeavouring to do so, we took a number of important technical decisions some of which are listed below:

- It was decided to keep ANURAG a lean establishment and not add infrastructure which was already available in industry. As a corollary, it was decided to take help from industry wherever possible. Thus it was that we forged a close alliance with ECIL which later proved beneficial to both of us.
- Next came the question of which microprocessor to use. Intel's 80386 was widely recommended to us and there was also the new hot favourite namely the Transputer, then widely hailed as THE solution that parallel computer designers were looking for. After much thought we settled on the Motorola microprocessor 68030, one of the important considerations being that our technical collaborator, i.e., ECIL had considerable experience with it.
- The machine itself was named PACE (Processor for Aerodynamic Computation and Evaluation) and the project goal set was the development of a 128-node

machine named PACE-128. It was agreed that PACE-128 would be arrived at by going through smaller machines namely, PACE-4, PACE-8, PACE-16, etc. This was intended among other things to give us experience in the scalability of the architecture.

- Although the Motorola microprocessor was selected for a start, it was decided that we would concentrate on a processor-independent design so that later when more powerful microprocessors became available, we could easily graduate to them. In retrospect, this proved to be one of our wisest decisions.
- Right from the beginning we decided to work in close association with the prospective users i.e., aerodynamists in the Aeronautical Development Agency (ADA), the organisation having overall responsibility for the LCA project. This also proved to be a very useful decision for we received invaluable user-feedback all through the development stage so much so when the final product was ready, it was pretty much what the user wanted.

ANURAG as an independent constituent unit of DRDO, which formally came into existence in May 1988. At that time our staff strength was barely a dozen, and this included administrative and support staff. But our unit was young and the members one and all were full of enthusiasm. Project PACE itself received formal sanction around August 1988 and funds were placed at our disposal.

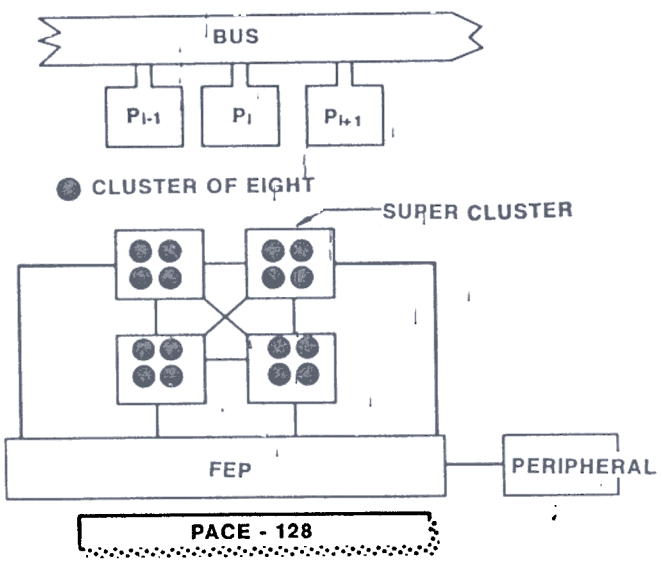

Figure 5. An overview of the architecture of PACE-128. Each cluster consists of eight nodes. Four clusters form a super cluster (SC). Four super clusters together with the FEP constitutes PACE-128.

We went full steam ahead but the going was quite rough. Apart from the usual teething troubles, power cuts and problems of that kind, we faced an unusual difficulty in the form of two long spells of curfew following the Ayodhya disturbances. What really saw us through was the unquenchable enthusiasm of our team.

By 1992 when the pilot project PACE was scheduled to end, we had assembled a 128-node system whose architecture is shown in Fig. 5. Although we had first started with the Hypercube concept, we found it prudent later to use the VME bus as a communication channel. Our basic unit had eight nodes plugged into a VME back plane. Four basic units suitably linked formed a cluster, and four clusters were linked into a super cluster as shown in Fig. 5 to realize PACE-128. The communication links between the cluster and a super cluster or between super clusters were via back plane extenders called VME-VME extenders. As mentioned earlier, we used the Motorola microprocessor 68030; for enhancing the speed, we backed the microprocessor by the Motorola coprocessor 68882.

PACE-128 worked exactly as we had expected it to but meanwhile microprocessor technology had galloped and the user now wanted much greater speed. But before I get to the question of speed, attention must be drawn to an important aspect of PACE, namely its software environment ANUPAM (ANURAG's Parallel Applications Manager), which makes the PACE machine quite user-friendly. Indeed, during the development phase we had scientists from many establishments coming to ANURAG to try out PACE (even though it was then in quite a raw condition). Almost all these people had no prior experience whatsoever in parallel programming and yet within a week at the most, they were able to use PACE quite comfortably.

6. TRYING TO GET SOME SPEED

Earlier, I had mentioned that we had chosen the Motorola 68030 processor for PACE-128, and to boost the speed, planned on using the Motorola coprocessor 68882. However, even at that time we were aware that 68882 would not give us the speed we really wanted. We therefore made the bold decision to design our own coprocessor which would be taster than Motorola's 68882. Many skeptics told us: "How is this possible? Motorola is a famous company with hundreds of

designers and so much experience. Do you think you can design a chip which is better than theirs?" We remained immune to such discouragement because we knew that a small company in the US had in fact designed a floating-point accelerator as a replacement for the Intel coprocessor 80387. If the Intel coprocessor could be outdone, then why not Motorola's? that was our thinking. Here I must mention that Dr Arunachalam had full confidence in us and backed us all the way.

Events proved that his faith in us was fully justified. Our accelerator named ANUCO (ANURAG's co-processor) was successfully designed in house but for fabrication we necessarily had to go abroad. There were numeric problems but eventually we made it; ANUCO worked exactly as expected i.e. three times as fast as 68882. By the way, ANUCO is quite a complex chip packing as it does over a hundred thousand transistors. It was then the most complex VLSI to be designed in India.

ANUCO's success was a wonderful boost to us, specially because an American company also had started trying to outdo the Motorola 68882 but gave up on the way. However, there were also some other factors which diluted our success. Firstly, thanks to delays (some of which were at the foundry end), ANUCO was delayed much more than it should have been. By the time the test pieces became available, technology had moved rapidly, leaving the Motorola 68030 + ANUCO combination rather far behind. Secondly, again due to dealing with an overseas manufacturer, there were numerous problems in arranging the quantity production of the ANUCO chip and this proved to be another setback. The moral of the story of course is that it is not enough to have just design capability in the country; one must in addition also have manufacturing support – and there is no reason why a giant country like India should not.

Notwithstanding these apparent failures, my own personal feeling is that the ANUCO experience was a success in that it firmly introduced into DRDO the notion of custom design of VLSIs.

7. PACE IN A NEW AVATAR

Although the project originally sanctioned by DRDO had formally came to an end in 1992 with the completion of PACE-128, the quest for higher speed did not come to a stop. Starting with Intel's *i*-860, a series of powerful microprocessors began to appear on the scene and it became clear that one could now easily get more speed with fewer nodes (see also Fig. 6). Keeping this in mind and utilising previous experience, ANURAG configured a new machine called PACE$^+$ with 32 nodes based on the Hyper Spare chip. This machine which has a speed approaching a gigaflop was formally inaugurated by the Prime Minister of India. PACE$^+$ now in ADA is working round the clock to the great satisfaction of the user.

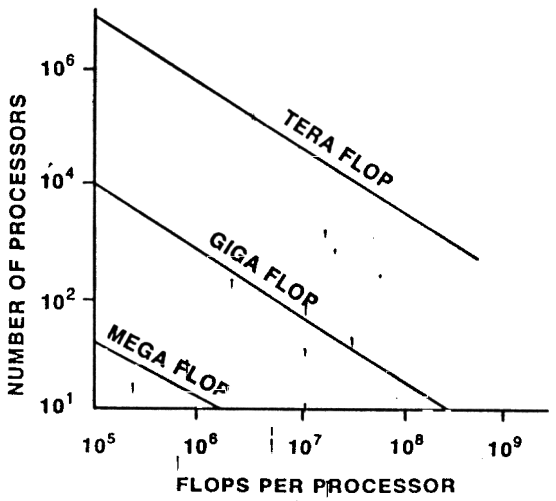

Figure 6. This figure indicates the relationship between number of processors required (for a specified total throughput) and the power of the individual processor or node. The greater the power of the microprocessor, the fewer the number of nodes required to achieve a specified throughput. The slant lines give the relationship when the throughput required is a Megaflop, a Gigaflop and a Teraflop. We saw this relation working in practice as we moved from our very first PACE-4 (based on Motorola 68030) to PACE$^+$ (based on the Hyper Sparc).

Right from the beginning we were conscious that there was not much use in building just one machine; rather, one must have a technology transferable to industry which then could supply machines to customers as and when needed. With this in mind ANURAG made suitable efforts to achieve transfer of technology thanks to which, many laboratories in the country now have PACE machines to suit their requirements and budget.

The goal of seeking eventual technology transfer imposed a good discipline on us during the project phase. It was not merely a question of bread-boarding a machine which would run whimsically and respond only to the magic touch of the designer. Our machine

would have to run anywhere, and be capable of operation by people who knew nothing of the design, and be capable of being maintained and supported by engineers of the company to which know-how was transferred. Above all, the system had to be robust and highly user friendly. It was good that we learnt to grapple with all these issues even during design. Coming to them as an afterthought would have considerably delayed the TOT in turn possibly reducing PACE to a mere research curio.

8. FOREIGN SCENE

It is pertinent to digress at this juncture and say something also about what has been happening abroad on the parallel computer scene. While all the early development work was in pure R&D establishments, commercial companies soon began to enter the picture with a variety of models. Among the users, particularly in America, the initial response to parallel computers was somewhat lukewarm especially since access was always possible to some CRAY machine or the other. "Who would be bothered about writing a program for a parallel computer? "that was the syndrome in the early nineties. However, views soon began to change and a good bit of the credit for bringing about this change goes to the massively parallel connection machine. A recent version of it, the CM 2 has 2^{16} = 65536 bit-serial processors which are custom-designed. The sixty thousand and odd processors are driven by a front end which provides the usual Fortran and C-computing

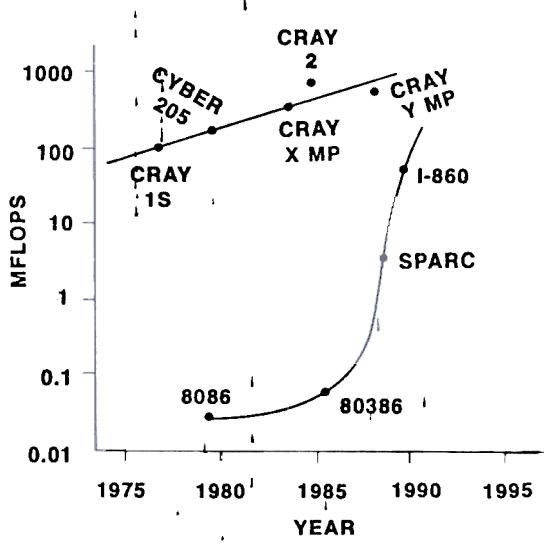

Figure 7. This shows how the growth in microprocessor power caught up with the CPU power of the CRAY machines. Along the ordinate is plotted the peak 64-bit floating-point performance.

environment. The front end could in fact be either a Sun or a VAX running the UNIX operating system. A number of very intensive problems have been executed on the CM 2, especially in the area of fluid dynamics, and this is what really opened the eyes of one and all in America to the great potentialities of the parallel computer.

This is not to suggest that the CM 2 holds a monopoly. As early as 1989, there were quite a few other machines on the market, and Table 1 (taken from Computers in Physics, Jan/Feb 1989) shows a list of some of the other machines then available in the US. Since then of course, things have changed even further on account of dramatic improvements in microprocessor capability, especially with the advent of the so-called RISC processor (Fig. 7).

What about the Japanese? Are they not into parallel computers? Of course yes! Historically, the Japanese developed their own super computers comparable to the CRAY and despite intense American

Table 1 Peak speeds of shared-memory multiprocessors and message-passing multicomputers (circa 1989)

Machine	Max. No. of proc.	Peak speed	Architecture
		188 MFLOPS	Shared-memory bus
Amtek series 2010	1024	20,000 MFLOPS	Hypercube
BBN butterfly	256	256 MIPS	Shared-memory switched network
Concurrent computer 3280	6	34 MIPS	Shared-memory bus
Convex C1 XP	5	25 MIPS	Shared-memory bus
CRAY Y-MP	8	6000 FLOPS	Shared-memory bus
ELXSI 6400 Model 2	12	156 MIPS	Shared-memory bus
Encore multimax 320	20	40 MIPS	Shraed-memory bus
Intel i-PSC/	128	424 MFLOPS	Hypercube
Masscomp 5700	4	52 MFLOPS	Shared-memory multibus
NCUBE/10	1024	500 MFLOPS	Hypercube
Sequent symmetry $ 81	30	80 MIPS	Shared-memory bus
Thinking machines CM2	65,536	20,000 MFLOPS	Hypercube

Source: Computers in Physics, January/February 1989

pressure Japanese R&D establishments always preferred the Japanese super computer to the CRAY (what a contrast to the Indian attitude!). At the end of 1991, about 150 vector super computers had been installed in Japan, representing about one-third of all the super computers in the world. In contrast to the US, most of the super computers in Japan had been acquired by industry. There is a lot of interest in fluid dynamics in the industry, for example in the design of high-speed trains. No wonder then that there is so much interest in number crunching amongst designers in Japanese industry. As in the case of vector super computers, the Japanese have decided to develop their own parallel computers, one example of this being the Fujitsu AP-1000, a massively parallel machine with up to 1024 processors. As a leading manufacturer of silicon chips, Fujitsu has also built a dedicated parallel computer called MAPLE with about 64,000 processors. This machine is meant for PCB and chip layout. Fujitsu believes the investment is very worthwhile as it would enormously cut down layout time, thus giving the company a price advantage when it comes to the sale of ASICs in the market.

One important aspect of Japanese philosophy is worthy of special comment. Observers and critics in the Western World frequently declare that whereas the Japanese might be more advanced in hardware, they are way behind America in software. Also, since the Japanese machines usually do not run all the exotic application software available on the Western market, the Japanese cannot really do much with all the great machines they are building. Events have proved this criticism to be false. True, the Japanese have not cared to port on their machines the codes which form the bread and butter of designers in America and Europe but the Japanese are not in the least bothered! Instead they have worked hard to develop their own brand of application software. To some this might appear naive. but Japan being a mighty industrial empire could well afford to support the development of its own application packages. Indeed with development subsidies, etc., local software is also cheaper for the industry; besides, there is the invaluable advantage of technology independence. In particular, Japanese industry now has the flexibility to move in the precise direction it wants, and not be steered by the capabilities of Western software.

One other issue which has cropped up recently deserves mention and that is the question of clusters versus tightly-coupled parallel machines. Thanks to rapid increase in the power of the RISC processor, workstations with CRAY-like capabilities (Fig. 8) are now increasingly becoming available at quite a low cost. Many scientists, especially in university environment, have considered using workstation clusters for running parallel programs. The question boils down to this: can a cluster of workstations effectively function as a parallel computer? In other words, could one user hog all the workstations and use the cluster just like a scientist at ADA would use PACE+? The short answer is yes, but it all depends very much on the characteristics of the application, the latency of data transfer (i.e., the lag between the times when a processor asks for data and when it can work on the data), the bandwidth of the communication channel etc. Clusters are limited to slower interconnection speeds and higher message latencies. As against this, there is the advantage of cost–clusters being cheaper in some cases and make sense especially in a university environment where there are always many workstations around, all networked in some fashion or the other.

While clusters might be popular in universities, hard core number crunchers in major R&D establishments like the Los Alamos National Laboratory (LANL) and the Lawrence Liver more National Laboratory (LLNL) have opted in clear favour of tightly-coupled parallel machines. In fact a cooperative agreement worth about $ 70 million has been signed involving the LANL, LLNL, the CRAY

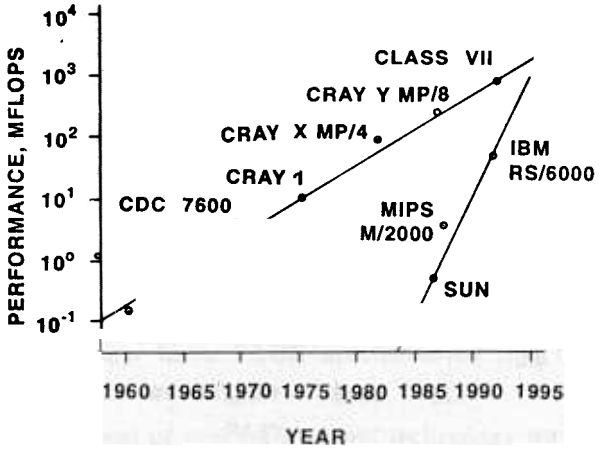

Figure 8. This shows how over the years, workstations have been sl wly catching up with super computers. It is this trend which has encouraged the cluster concept.

company and the US Department of Energy, aiming to support the development of powerful MPPs (massively parallel processors), and to popularise the use of parallel computers. As a part of this agreement, CRAY has now entered the parallel computer field and has come up with T3D, a parallel computer based on the powerful microprocessor the DEC Alpha. The first machines have already been delivered. CRAY expects that by 1997, its machine (which could involve as many as 2048 processors) would be able to deliver 1 TFLOPS (TFLOPS = trillion FLOPS) of sustained performance.

While impressive strides have been made on the hardware side, there are many issues still to be settled where software is concerned. Two questions which have arisen are:

1. What about the portability of programs across machines offered by different vendors? Suppose a scientist has developed a program which runs well on a particular parallel machine. Say his laboratory acquires a new machine offered by another manufacturer. Can the program be easily run on the new machine or would it require an enormous affort to adapt? Now that the Western world is entering the second generation as far as parallel machines are concerned, questions of this type are assuming importance.
2. Many codes exist which have been the workhorses for decades (this is particularly true of the aircraft companies in the US and of the nuclear weapons laboratories there). Can one have an automatic parallelising compiler which would, without any sweat to the scientist, efficiently parallelism the serial code of an earlier era?

These questions cannot be answered in this article for they lie far beyond its scope. I have presented them merely to highlight some of the current global trends in R&D in parallel computers. In general, one can see a phase where pioneers are being replaced by settlers. Standards for parallel computer software will also soon become an important issue (it they have not already done so), and this too is one of the things to watch for.

9. SOME REFLECTIONS

Before concluding, I would like to indulge in a few reflections. Firstly, I would like to make it clear that the account I have given here about the development of parallel computers in India refers strictly to the effort mounted in DRDO. Just about the time we started our work, several other groups also commenced similar developmental activities. So the question has often been asked: "Why so many parallel efforts in developing parallel computers? Can the country afford this duplication?" Good question.

The question whether duplication is desirable or otherwise depends very much upon the resources that need to be invested, both financial and human. If, for example, one considers the development of an aircraft, clearly India can afford only *one* well-coordinated R&D effort. (By contrast, the US which aims at being a leader in aerospace technology and which has the resources, supports a multiple development approach). In the case of parallel computers, however, the investments involved are quite minor. People often tend to feel that a few crores spent on developing technology is too much of a waste but none gives a second thought to the fact that today, a much larger sum is being spent on VIP security. So, it really boils down to our concept of priorities. Spending money in a calculated fashion on the development of frontier technology is *not* a waste; it becomes a waste only if the exercise is not properly planned and coordinated.

I personally feel that it was quite worthwhile for the country to have supported several developmental efforts in the area of parallel computers. May be some efforts did not do so well in purely clinical terms but in technology, even failure is a valuable lesson. More important, many more people got exposure and equally relevant, many avenues of solution to the problem got studied. It is only through such trial and error that consolidation, maturity, etc come. Thus, in terms of the intangible called acquisition of expertise and human resource development, the investment was certainly worthwhile. In the nuclear field with which I have been associated earlier, I have seen this happen on a global scale. With regard to nuclear power, several reactor concepts were tried out in the early days—the light water reactor, the heavy water reactor, the graphite reactor, etc. It was only as the result of such multiple experimentation that consolidation could eventually be achieved. One might still argue: "All this is fine for the rich countries but not for India". And my reply would be: "Yes when it comes to truly expensive projects (like the space project, for example) but not in cases like the development of electronic systems."

I believe that the question we should really be asking is not whether we can afford multiple development (which reflects an auditor's approach) but whether we are getting all the benefits of multiple development that we ought to? Unfortunately, this question is never asked and that is where the problem lies.

One may ask: "What sort of benefits is one talking about?" And the answer would be the following:

- Parallel computers are here to stay. In the future, all big problems would necessarily have to be done on parallel computers, be it aircraft design, VLSI design, or even the safety of engineering structure like automobiles for example. Now that several parallel computers have been developed in the country, has any special effort been mounted to make our scientific programmers conscious of parallel computing and how to write programs for parallel computers? One should here take note of the conscious thrust being made in America, jointly by the national laboratories, industries and government. Awareness about parallel computing and acquisition of expertise in that would be one of the benefits.
- We went through Phase I. What about a follow-up? Elsewhere this is happening; is our effort of the past to vanish without a trace like a river in desert? In short, Phase I should have led to a properly coordinated Phase II, and if that has not happened, then it means that we are again losing the benefit of what was gained earlier.

COMPUTING ALTERNATIVES

	FLEXIBLE MULTIPROC.	MAINFRAME SUPERCOMP.	PARALLEL SUPERCOMP
SCALABLE TO VERY HIGH PERFORMANCE	✓	—	✓
LOW COST	✓	—	✓
TRADITIONAL PROGRAMMING MODEL	✓	✓	—

Figure 9. This figure offers at a glance the various computing alternatives. The options available thus far are broadly represented by the mainframe super computers (e.g. CRAY) and by the parallel super computers. Soon one may expect flexible multiprocessor supercomputers, which combine the best of both worlds.

- The parallel computer was introduced to deal with intensive number-crunching problems. Typically, all the processors in the system, even if there were a thousand or more, were pressed into one problem posed by one user. On the other hand, in the commercial world, one has for some years now been hearing about multiprocessor systems. Are there applications where one need both parallel processing as well as multiprocessing capabilities? Indeed there are many. To give a simple example, a hospital may have complex diagnostic equipment involving medical imaging. Advanced application in 3-*D* visualisation benefits greatly from parallel processing. On the other hand, a hospital would typically have thousands of image data files which necessarily have to be managed by a database kind of a system. Thus, one could conceive of a large and flexible computer with many processors, part of which functions like a parallel processor (on computationally intensive problems), and part on data management.
- To put it in more technical language, what one is talking about is a computer system which combines the very high level of performance and low costs of computation inherent in parallel processing with an environment more suited for commercial and business applications, like online-transaction processing (OLTP). The place of the proposed hybrid *vis-a-vis* previously known systems is illustrated in Fig. 9. Machines such as these will be immensely useful in factories. stock exchanges etc. Flexible multiprocessor systems could thus potentially be very important for us, and may be we should have made a foray into this field after the completion of Phase I. The task is also very challenging, which is another reason why we should have given some attention to it.

Presentation graphics coupled to parallel computers is yet another area which should be attracting our attention. Especially in CFD (computational fluid dynamics), visualisation is exceedingly important. Most of the parallel computers of today do not have built-in graphics capability. Building this would in my opinion be very useful and this too should be engaging our attention.

The above, I trust, is enough to give a flavour of the kind of introspection we should be doing. I shall desist from adding further. Perhaps this is the right juncture to comment on the "drought" referred to earlier.

There was no encouragement or incentive to write large and innovative programs. Globally, all the major advances through the use of computer simulation have

come from large application programs. Few programs of this kind were written in India and whenever the need for them became pressing, people always tried to get them from abroad, hardly appreciating that what is sold is only the executable module and that we *do not* get the source code, without which further extensions are not possible. And because it was that over a decade or so, we learnt to become competent drivers instead of designers. Having got into that habit (of driving imported programs), we now seek to perpetuate it. Those involved with technology development aid the process by saying: "Sorry, I don't have all the time you need for code development. I have a project deadline to meet. So let us buy the code from abroad." In many cases this does not work, for application codes are what technology is all about, and they would not be sold for love or money. This is true, for example, in the aerospace industry. Japan has always been conscious of this and has therefore heavily supported indigenous code development. We too need such development because often the codes even if available, are frightfully expensive. But people have learnt to carry on without writing codes. And in every area from oceanography to computer graphics and image processing, people are clamouring for imported packages. If, as is often claimed, we have a huge pool of software expertise, why on earth must one talk of imports all the time?

To sum up, thanks to a brief visionary wave that swept the country in the late eighties, many of us, myself included, had the wonderful opportunity to work in a frontier area of technology. For me personally, this was a refreshingly different and a most satisfying conclusion to a long career which earlier had revolved entirely around physics. While I still love physics, this country needs technology even more than it needs physics and I am really glad I could play a small role in technology development, even if it was only as a manager rather than an innovator. On the other hand, I could ring the curtain down not only having groomed many a young person. but having myself picked up something in computer applications. Indeed, in my post-retirement period. one of the subjects which I am teaching is what I learnt in ANURAG, namely digital image processing.

Quantum jumps occur either through the vision of one man or ruthless planning by hard core group (like the MITI in Japan). We generally seem to lack both; one hopes better times will soon be with us again for without either vision or hard planning, we may drift even more in this new so-called liberalised economy.

10. ACKNOWLEDGEMENTS

I owe a deep debt of gratitude to all the members of ANURAG for their wonderful cooperation which enabled the Group to complete the PACE project, particular thanks going to Dr K Neelakantan, the Project Leader and his able associates G Athithan, P Ghosh and K Santeppa. I would also like to place on record my personal indebtedness to Dr V S Arunachalam for the enormous confidence he reposed in our team. ANURAG is really his baby. I am equally indebted to Dr A P J Abdul Kalam, SA to RM, for the invaluable support he gave first as Director, DRDL, Hyderabad and later in his present capacity. If ANURAG has blossomed in recent years, it is in no small measure due to Dr Kalam.

Contributor

Prof G Venkataraman is Vice-Chancellor of Sri Sathya Sai Institute of Higher Learning (Deemed University). Prasanthi Nilayam, Andhra Pradesh.