Notes on numerical fluid mechanics

Willi Schönauer
Wolfgang Gentzsch *Editors*

The efficient use of vector computers with emphasis on computational fluid dynamics

A GAMM_Workshop

Willi Schönauer
Wolfgang Gentzsch (Eds.)

The Efficient Use of Vector Computers with Emphasis on Computational Fluid Dynamics

Notes on Numerical Fluid Mechanics Volume 12

Series Editors: Ernst Heinrich Hirschel, München
Maurizio Pandolfi, Torino
Arthur Rizzi, Stockholm
Bernard Roux, Marseille

Willi Schönauer
Wolfgang Gentzsch (Eds.)

The Efficient Use of Vector Computers with Emphasis on Computational Fluid Dynamics

A GAMM-Workshop

Springer Fachmedien Wiesbaden GmbH

Produced by Lengericher Handelsdruckerei, Lengerich

ISBN 978-3-528-08086-0 ISBN 978-3-663-13912-6 (eBook)
DOI 10.1007/978-3-663-13912-6

PREFACE

The GAMM Committee for Numerical Methods in Fluid Mechanics organizes workshops which should bring together experts of a narrow field of computational fluid dynamics (CFD) to exchange ideas and experiences in order to speed-up the development in this field. In this sense it was suggested that a workshop should treat the solution of CFD problems on vector computers. Thus we organized a workshop with the title "The efficient use of vector computers with emphasis on computational fluid dynamics". The workshop took place at the Computing Centre of the University of Karlsruhe, March 13-15,1985. The participation had been restricted to 22 people of 7 countries. 18 papers have been presented.

In the announcement of the workshop we wrote: "Fluid mechanics has actively stimulated the development of superfast vector computers like the CRAY's or CYBER 205. Now these computers on their turn stimulate the development of new algorithms which result in a high degree of vectorization (scalar/vectorized execution-time). But with 3-D problems we quickly reach the limit of present vector computers. If we want e.g. to solve a system of 6 partial differential equations (e.g. for u, v, w, p, k, ε or for the vectors $\underline{u}$, $\underline{\text{curl } u}$) on a 50x50x50 grid we have 750.000 unknowns and for a 4th order difference method we have circa 60 million nonzero coefficients in the highly sparse matrix. This characterizes the type of problems which we want to discuss in the workshop".

When the first CRAY-1 (12,5 nsec cycle time) was delivered in 1976, it was an "exotic", a "supercomputer". It seemed that only a very few large research establishments would be able to afford such a computer. Now, roughly 9 years later, the CRAY-1 is history, it has been replaced by the CRAY X-MP (9,5 nsec cycle time) and the CRAY-2 is now on the market (4 nsec cycle time). But the most unexpected thing in this development is, that circa 70 CRAY-1's have been delivered. From the other "early" vector computer which reached commercial significance, the CYBER 205 of CDC (20 nsec cycle time), circa 30 units have been delivered. In Germany in the field of research there are presently installed 4 vector computers at universities and 3 vector computers in research establishments. In the USA a program has been initiated to give access to all universities to vector computing centres. In the commercial sector all large industrial companies in the field of aircraft, automobile or oil will have one or several vector computers if they do not have already.

Ultimately the extreme computational speed of the vector computers results from the parallelism which is inherent in most largescale computations (mostly this is some form of matrix manipulation). But the programmer has to write his programs in such a form that the vector computer is able to transform the language instructions into vector operations, otherwise it will compute only in scalar mode, i.e. like a usual general purpose computer. But we are still on the way to learn this "parallel computing". In this sense the goal of the workshop was to put together the experiences of the participants that they might present their own experiences to the other participants and, on the other hand, learn themselves from the experiences of the others.

We have mentioned above that a 3-D grid of 50x50x50 is the limit for the present generation of vector computers. But for the description of the flow around an automobile or a whole aircroft in sufficient details a grid of 500x500x500 would be needed. This requires the 1000-fold computational speed of the present generation of vector computers. If we denote the first generation by the "100 MFLOPSgeneration" (MFLOPS=million floating point operations per second), we have now by the recently announced CRAY-2 the 1 GFLOPS (GFLOPS= 1000 MFLOPS), around 1988 10 GFLOPS and perhaps 1992 the 100 GFLOPS. This will be possible only by a new type of parallelism: the multiprocessor vector computers. But these speeds can be achieved only if the main memories are large enough or if there are fast and large enough secondary storages to store the immense data sets of such large problems. The development of superfast computers will surely never end. Neil Lincoln who has designed the CYBER 205 and who is now designing the successor, the ETA^{10}, gave the following definition of a supercomputer: "A supercomputer is a computer which is only one generation behind the requirements of the large scale users". These remarks may conclude the definition of the background of the workshop.

The papers of the workshop are not presented in these proceedings as usual in alphabetic order of the authors. We had composed the program of the workshop by grouping the papers according to the subjects which were treated, to the vector computers and type of solution methods which were applied. And in the same sequence the papers are presented in these proceedings. We decided also that the participants should write their papers immediately after the workshop and that they should include suggestions which were obtained from the discussions. In this way, most recent results have been included.

In an introductory paper the hard- and software of the most relevant vector computers with their typical properties and deficiencies are presented. Then a group of 6 papers deals with the full potential and Euler equations. It is very informative to see how these problems are treated on the different types of vector computers by different types of discretization methods. Then follows a paper which treats the shallow water equations. The solution of the 2-D and 3-D Navier Stoker equations is presented in a group of 6 papers. Then follow two papers of the field of meteorology and climate modelling. A further paper reports on experiences with the testing of different linear algebra algorithms on vector computers in the frame of CFD problems.

In the final paper which was ultimately worked out during the workshop a summary and a conclusion of the workshop are presented. This paper tries to classify the problems which were treated on the different vector computers and the methods which were used, states the problems and points out the direction which further developments should take. We hope that these proceedings of the workshop will enable the reader to profit in the same way as we did from the cumulated experience which was gathered at the workshop.

June 1985 Willi Schönauer Wolfgang Gentzsch

CONTENTS

INTRODUCTION TO THE WORKSHOP: SOME BOTTLENECKS AND DEFICIENCIES OF EXISTING VECTOR COMPUTERS AND THEIR CONSEQUENCES FOR THE DEVELOPMENT OF GENERAL PDE SOFTWARE

W.Schönauer, E.Schnepf

Rechenzentrum der Universität Karlsruhe

Postfach 6380, D-7500 Karlsruhe 1, West-Germany

SUMMARY

In the first part of this introductory paper a review of the different types of vector computer programs and algorithms is presented. Then the hardware of the presently most relevant vector computers together with their bottlenecks will be discussed as well as the trends in development. The weakest point of the existing vector computers is the compiler, this problem is closely related to the lacking vector statements of FORTRAN 77. Therefore some proposals of FORTRAN 8X will be presented. In the second part of this paper the authors report about some experiences which have been obtained in the development of "black-box software" for PDE's. There will be given a pragmatic definition of a "data flow algorithm", and the separation of data selection and processing will be demonstrated for the evaluation of difference formulae. Then the i/o bottleneck is discussed for the iterative solution of large linear systems in diagonal storing. There will be presented our view of portability of software for different vector computers. The numerical example for ILU-preconditioning demonstrates how vectorisation might invert experiences gained on general purpose computers. Finally some examples demonstrate the application of the FIDISOL program package on different vector computers.

1. HISTORICAL DEVELOPMENT OF PROGRAMS AND ALGORITHMS

If we consider in a sort of historical review the development of <u>user programs</u> on vector computers, we might distinguish between 3 generations of vector computer software:

1.1 Unchanged general purpose computer programs. These programs have been developed on general purpose computers for general purpose computers. They have many nonvectorisable loops and a bad data structure. On present vector computers they obtain (eventually) 10-15 MFLOPS (million floating-point operations per second).
1.2 "Hand vectorised" general purpose computer programs. The user tries to remove, mostly on the level of the coding, the reasons for the nonvectorisation of the loops, using the information of the compiler who tells to the user which loops why have not been vectorised. But usually there remains still the bad data structure. Such programs may obtain 15-20 MFLOPS.
1.3 Special vector computer programs. These programs have been designed from scratch for vector computers, with an optimal data structure and with optimal algorithms. Such programs may obtain a sustained rate of 40-80 MFLOPS.

In a similar way we might distinguish between three generations of vector computer algorithms:
2.1 Unchanged general purpose computer algorithms, with many recursions, with short vector length, not designed for vectorisation.
2.2 "Vectorised" algorithms, still incore, only designed to solve the old 2-D or explicit 3-D problems very fast, but they are not suited for large data sets.
2.3 Special vector computer algorithms, designed for large 3-D data sets, implicit solution methods, optimal data structure also for out-of-core. The performance of such algorithms is not limited by the speed of the vector pipes but rather by the i/o bottleneck of most installed vector computers.

The conclusions which we can draw from this simplifying historical review of user programs and algorithms are the following ones:

- "Old" general purpose computer software wastes the power of a vector computer. This danger holds above all for large industrial program packages into which have been invested many many man years of programming. People who judge the power of a vector computer from the generations 1.1 or 1.2 mentioned above, may be well disappointed. It might be interesting to note that in comparison to a CYBER 205 a CRAY-1 usually will deliver shorter execution times for such programs because the CRAY-1 is not so "sensitive" to "bad" software.

- The power of a vector computer grows considerably with the skill of the user, much more than would be possible on a general purpose computer. Going from 10 MFLOPS for "old" software to 80 MFLOPS for special vector computer software means an increase of the huge power of a vector computer by a factor of 8! It is interesting to note that for special vector computer software of the generation 1.3 the CYBER 205 usually will win over the CRAY-1, above all if the algorithms are designed for long vectors.

- The increased speed of special vector computer software immediately allows to attack larger problems, but these problems then produce larger data sets.

- Large 3-D problems (and the real world is 3-D!) are on the present generation of installed vector computers more severly limited by the i/o than by the arithmetic speed of the vector pipes because the size of the main storage is usually too small to allow an incore treatment of 3-D problems. This misbalancing results from the extremely expensive technology which was used in the main memories.

- An efficient use of a vector computer is guaranteed only by "data flow" type algorithms which maintain a continuous data flow through the pipes.

- Therefore we need for vector computers well trained users, who must be much better trained than for general purpose computers.

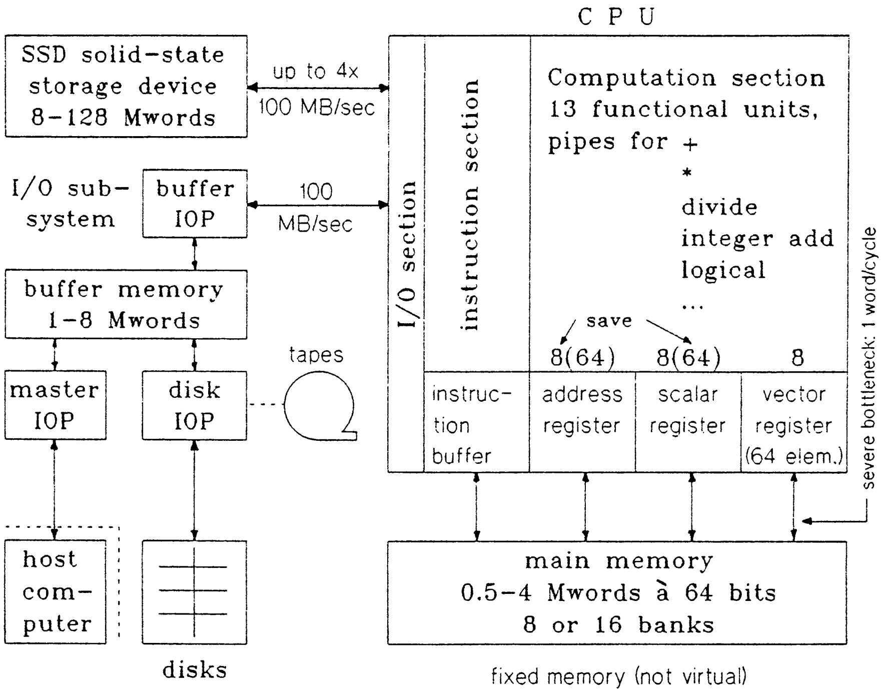
- It might be useful at this point to mention the literature which can help to train our users better. There is the excellent book of Hockney/Jesshope [1] which is as a general introduction to vector computers up to now the best reference. The book of Gentzsch [2] is a natural extension of the introductory literature as it contains many useful applications and is in some sense similar to the intentions of this workshop.

2. THE HARDWARE OF THE MOST RELEVANT VECTOR COMPUTERS

2.1 CRAY-1

The CRAY-1 has a cycle time of 12.5 nsec (nano sec = 10^{-9} sec), it has $n_{1/2} \approx 20$. Hockney's $n_{1/2}$ is the vector length which is needed to get half the real peak performance [1]. The CRAY-1 is now history, it is no longer produced. Between 60 and 70 units have been delivered. The CRAY-1 is a milestone in the development of vector computing. It started as an "exotic" computer and then succeeded in making the breakthrough for commercial vector computing. From all the vector computers mentioned in this paper the value $n_{1/2} \approx 20$ is the smallest one. This has the consequence that programs need not be designed for long vectors. Thus also programs of the generations 1.1 and 1.2 of section 1 run "efficiently" on the CRAY-1.

The CRAY-1 is the prototype of a register-to-register machine: The functional units (pipes) operate only with operands which are stored in one of the 8 vector registers (each V-register has 64 elements). This is the great chance and great danger of this type of computer at the same time: The efficiency depends largely on the chaining, i.e. the overlapping of



CPU
SSD solid-state storage device 8-128 Mwords
up to 4x 100 MB/sec
I/O sub-system
buffer IOP
100 MB/sec
buffer memory 1-8 Mwords
tapes
master IOP
disk IOP
host computer
disks
I/O section
instruction section
Computation section 13 functional units, pipes for + * divide integer add logical ...
save
8(64)
8(64)
8
instruction buffer
address register
scalar register
vector register (64 elem.)
severe bottleneck: 1 word/cycle
main memory 0.5-4 Mwords à 64 bits 8 or 16 banks
fixed memory (not virtual)

Fig. 2.1 Schematic view of the CRAY-1 S.

operations and register loading. If all operands are in the vector registers, the operations are "uncoupled" from the main memory. But if the operands are not present, the pipes have to wait. It is obviously still a severe problem for the compiler to do an efficient chaining.

The CRAY-1 has a severe bottleneck: There is only <u>one</u> word transfer/cycle between main memory and V-registers <u>to</u> <u>or</u> from the main memory. But a simple operation like

$$c_i = a_i + b_i \tag{2.1}$$

needs 3 transfers/cycle, two loads and one store, but there is possible only one transfer/cycle. Therefore for such simple operations the peak performance is not limited by the 80 MFLOPS of the arithmetic pipe but by 80/3 ≈ 27 "MFLOPS" of the data path between main memory and V-registers. This bottleneck has also the consequence that long vectors are subdivided by the hardware into sections of 64 elements and after each section of 64 elements there is a restart of the pipe, leading to a saw-tooth curve for the MFLOPS-rate. It is possible to process noncontiguous vectors with constant stride, but this bears the danger of bank conflicts. The CRAY-1 S has a bipolar memory with 4 cycles bank-busy-time and the CRAY-1 M has a MOS memory with 8 cycles bank-busy-time. If a bank is addressed befo-re its bank-busy-time is over there are lost cycles for wai-ting. Thus also on the CRAY-1 the use of contiguous vectors should be recommended to avoid bank conflicts. Without the fast secondary storage SSD there is a severe i/o bottleneck, i.e. the data path between main memory and secondary storage is too narrow compared to the speed of the vector pipes, leading to problems for out-of-core programs. **There are no hardware in-structions for gather/scatter which are needed for all indirect addressing problems.**

2.2 CRAY X-MP/2

The denotation /2 means two processors. The cycle time of the CRAY X-MP is 9.5 nsec, it has a value $n_{1/2} \approx 50$ which is already 2.5 times that of the CRAY-1. The bottleneck

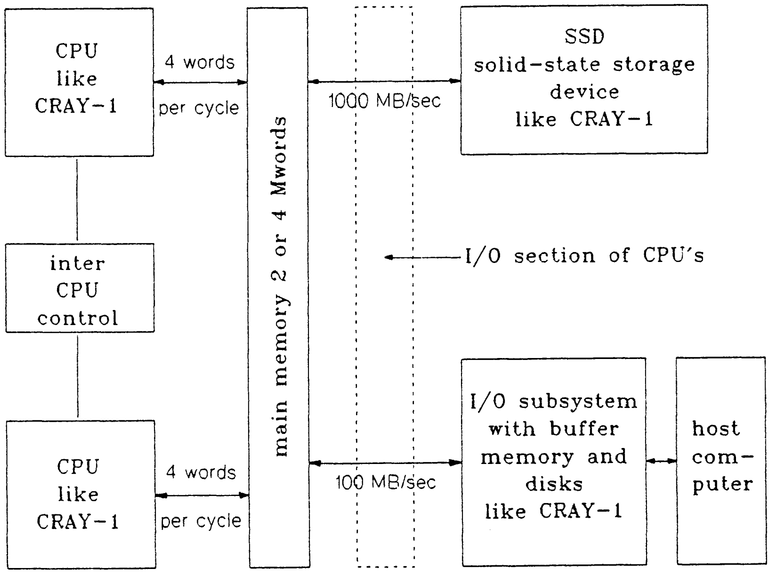


Fig. 2.2 Schematic view of the CRAY X-MP/2. The SSD and the i/o subsystem are in reality connected to the i/o-section of the CPU's.

between main memory and V-registers has been removed: There are for each processor 3 transfers/cycle between main memory and V-registers and one transfer/cycle for i/o. Although the vector length of each V-register is 64 elements there might be a con-tinuous data flow from the main memory through the V-register to the pipe and back to another V-register and from there to main memory that there is not a restart after 64 elements as in the CRAY-1. Although theoretically the bottleneck between main memory and V-registers has been removed one has to be ca-reful. If we consider a X-MP/22, i.e. with two processors and two Mwords (2 million words) main memory, the main memory has 16 banks and because of the 4 cycles bank busy time there are possible only 16/4 = 4 transfers/cycle. If we assume that each processor executes the simple operation (2.1) there are needed 6 transfers/cycle. But because there are possible only 4 transfers/cycle the peak rate would not be determined by the 105.3 MFLOPS of the pipes but by the $105.3 \cdot 6/4 \approx 70$ "MFLOPS" of the main memory modularity. Only the X-MP/24 with 4 Mwords and 32 banks has enough modularity of the main memory.

The CRAY X-MP/1 has only one processor and a MOS main mèmory with 8 cycles bank-busy-time. It has now replaced the CRAY-1.

The CRAY X-MP/4 has 4 CPU's and 8 Mwords main memory. It can be equipped with two 1000 MB/sec channels to the SSD. The CPU's of the X-MP/4 (only!) have now also hardware instructions for gather/scatter and compressed index instructions. This means that this computer will behave much more efficient e.g. for finite element calculations which needed in an extensive manner indirect addressing.

With the X-MP/2 and X-MP/4 CRAY makes a first step into the direction of a MIMD (multiple instruction stream / multiple data stream) computer. There are special registers for inter CPU control to exchange synchronisation information. CRAY offers software that several processors can work for one single job, i.e. for multiprocessing, which they call "multitasking".

2.3 CYBER 205

The CYBER 205 of CDC (Control Data Corpor.) has a cycle time of 20 nsec. The computer can be purchased with 1 or 2 or 4 vector pipes and the corresponding values of $n_{1/2}$ are 50 or 100 or 200. This means that for a 4-pipe CYBER 205 the vector length to get half the peak performance is 10 times as large than for the CRAY-1, i.e. such a computer can be efficiently used only with long vectors. A bit of history: Another mile stone in the development of vector computers was the STAR 100 which taught us that a vector computer without a fast scalar unit is not efficient. Equipped with a fast scalar unit it became the CYBER 203 and again equipped with faster vector pipes it became the CYBER 205.

The CYBER 205 is the prototype of a memory-to-memory machine: The vector pipes operate on data from the main memory to the main memory, the stream unit acting as an operand buffer. Vectors must be contiguous storage locations in the main memory. The main memory has the huge virtual address space of $2 \cdot 10^{12}$ words. But the computer has a severe drawback: The maximal (hardware) vector length is 64K-1 = 65535 (what a pity!) which is even not hidden by the compiler. The reason is the 16 bits for the vector length register. The computer has separate pipes for scalar and vector arithmetics which can operate in parallel for independent operations. A great advan-

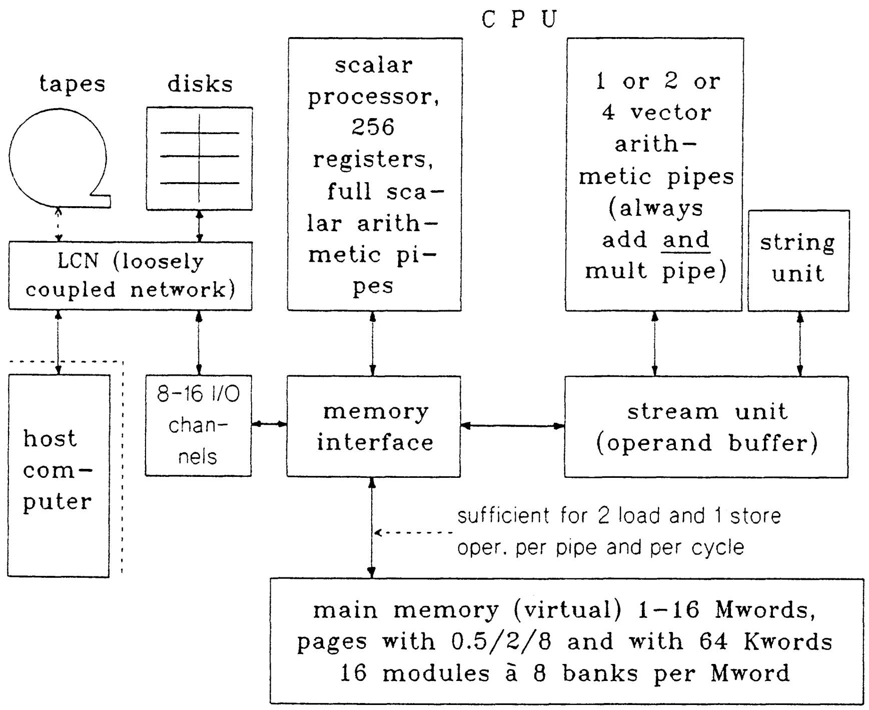
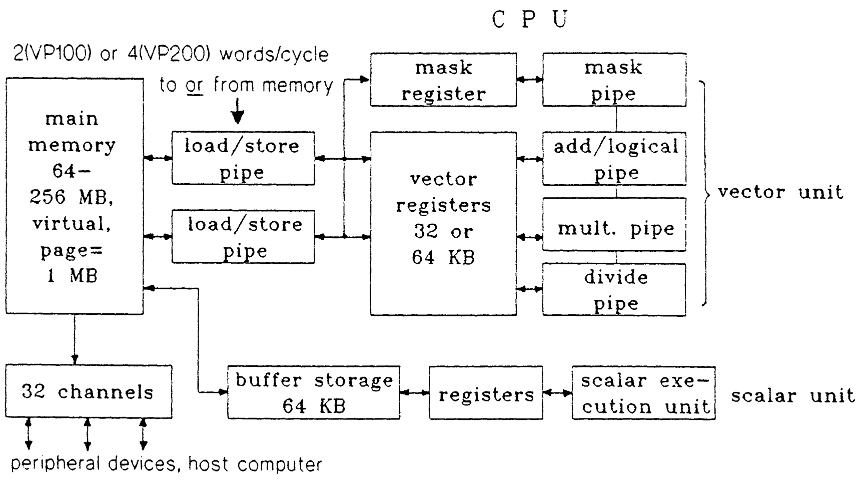

C P U
tapes
disks
scalar processor, 256 registers, full scalar arithmetic pipes
1 or 2 or 4 vector arithmetic pipes (always add and mult pipe)
string unit
LCN (loosely coupled network)
host computer
8-16 I/O channels
memory interface
stream unit (operand buffer)
sufficient for 2 load and 1 store oper. per pipe and per cycle
main memory (virtual) 1-16 Mwords, pages with 0.5/2/8 and with 64 Kwords 16 modules ã 8 banks per Mword

Fig. 2.3 Schematic view of the CYBER 205.

tage of the CYBER 205 are the sophisticated hardware instructions for gather/scatter, compress/expand/merge which allow for a relatively efficient indirect addressing. There is no fast secondary storage available, thus there is a severe i/o bottleneck. The CYBER 205 has the possibility of 32-bit arithmetic with twice the speed of the 64-bit operations.

2.4 FUJITSU VP 100 and VP 200

The VP 200 has "internally doubled" pipes, i.e. parallel pipes compared to the VP 100. The cycle time is 15 nsec for the scalar unit and 7.5 nsec for the vector unit. The value of $n_{1/2}$ is not yet available, but because of the parallel pipes $n_{1/2,\mathrm{VP}\ 200} = 2 \cdot n_{1/2,\mathrm{VP}\ 100}$. The Japanese have developed the basic technology in a common research project between the hard competitors Fujitsu, Hitachi and NEC and thus have saved a considerable amount of development costs. They have combined the advantages of the CRAY and CYBER vector computers. We consider this a "fruitful" competition which will force the "Americans" to develop better and faster vector computers and thus speeds up the development.



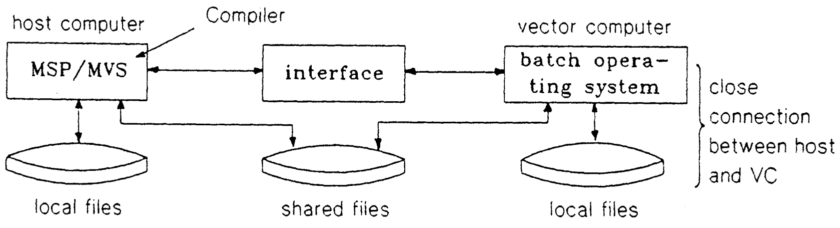


Fig. 2.4 Schematic view of the Fujitsu VP 100/200.

The VP 100 or 200 has also a bottleneck between main memory and vector registers. For the VP 100 resp. 200 only 2 resp. 4 words/cycle can be transfered, but for simple operations like (2.1) there are needed 3 resp. 6 transfers/cycle, i.e. there is available only 2/3 of the necessary bandwidth. It is interesting to note that the vector register file can be combined dynamically to a few long or many short registers. There is a close connection between host and vector computer: The compiler runs only on the host computer and on the VP there are executed only the load modules.

Hitachi has developed on the same basic technology his S810/10 and S810/20 and NEC will soon deliver its SX-2 with a cycle time of 6 nsec, i.e. with an enhanced technology.

2.5 BOTTLENECKS

2.5.1 Between main memory and pipes

This bottleneck is closely related to the notion of supervector speed: To keep busy more than one pipe, i.e. to obtain more than one result per cycle time. The bottleneck may be caused by the lacking data paths to the main memory or by the lacking modularity of the main memory. It can be reduced for

2.5.1.1 the register-to-register machine by the chaining. If it is possible to overlap loading, execution and storing one can "uncouple" from the main memory and e.g. the add and mult pipes may be busy in parallel without (excessive) memory references.

2.5.1.2 the memory-to-memory machine CYBER 205 by the linked triad, e.g. for

$$c_i = a_i + s \times b_i \, , \qquad (2.2)$$

i.e. a triadic operation with one scalar operand s. There are needed only the available two load and one store operation from and to the memory. Then the add and mult pipes contained in each pipe of the CYBER 205 can be coupled together and work in parallel. Thus the linked triad is the "only chance" for supervector speed for the CYBER 205. If there would be a third load-path, then for the full vector triad also supervector speed could be obtained.

2.5.1 i/o-bottleneck between main memory and secondary storage units

For large 3-D problems not all the data fits into the main memory. Then in an out-of-core version of the program operands must be fetched from the secondary storage. In order to demonstrate the i/o-bottleneck we assume that in the simple operation (2.1) one of the two operands e.g. a_i must be fetched from the secondary storage and that b_i and c_i reside in the main memory. Then we can transform the measured MFLOPS-rate into a necessary transfer rate, neglecting the access times. The resulting transfer

Table 2.1 Measured MFLOPS-rates and corresponding transfer rates for one operand.

Computer	CRAY-1	CRAY X-MP one processor	CYBER 205 1-pipe	CYBER 205 2-pipe	CYBER 205 4-pipe
measured peak MFLOPS for +,× for n=10 000	23.4	67.5	49.7	98	197
corresponding bits/sec for one operand	$1.5 \cdot 10^9$	$4.3 \cdot 10^9$	$3.2 \cdot 10^9$	$6.3 \cdot 10^9$	$12.6 \cdot 10^9$

rates can be seen in Table 2.1. Now we compare these transfer rates to those of a single disk and for parallel i/o by 10 disk channels:

disk IBM-type 3350: $3.5 \cdot 10^7$ bits/sec, 10 disks: $3.5 \cdot 10^8$ bits/sec,
3380: $8.1 \cdot 10^7$ bits/sec, 10 disks: $8.1 \cdot 10^8$ bits/sec.

If we compare these transfer rates, even for 10 parallel disk channels, we can see that they are not sufficient, and we have still neglected the access times. Thus disks are an insufficient secondary storage medium for a vector computer for working data sets.

If we now take as secondary storage medium the SSD (Solid -state Storage Device) of CRAY and take the maximum number of channels we can get the following transfer rates:

X-MP/1	100 MB/sec	= $8.0 \cdot 10^8$ bits/sec,
CRAY-1	4×100 MB/sec	= $3.2 \cdot 10^9$ bist/sec,
X-MP/2	1000 MB/sec	= $8.0 \cdot 10^9$ bits/sec,
X-MP/4	2000 MB/sec	= $16.0 \cdot 10^9$ bits/sec.

Comparison with Table 2.1 shows that we cannot meet the required transfer rate for the X-MP/1 because of only one channel of 100 MB/sec. But for the other computers, the CRAY-1 and the X-MP with 2 or 4 processors we can meet the requirements of Table 2.1. We shall come back to the question of the i/o-bottleneck in section 4.3.

2.6 VISIBLE TRENDS IN DEVELOPMENT

The third Japanese vector computer, the NEC SX2 will be officially announced during the printing of these proceedings. It will have 6 nsec cycle time and will reach by 4-stream vector pipes (results in large $n_{1/2}$!) a peak rate of 1.3 GFLOPS (gigaflops = 10^9 floating point operations per second). A first report about the basic concept can be found in [3].

CRAY will announce officially also during the printing of these proceedings its CRAY-2, with 4 nsec cycle time and 4 processors which are equipped like the CRAY X-MP/4 with hardware instructions for gather/scatter, compressed index instructions etc. The peak performance will be 2 GFLOPS. The main memory will have 64 to 256 Mwords (!!) which will allow to treat really large problems incore. Later versions should have up to 16 processors. The operating system will be UNIX and not COS.

CDC has founded a new Company ETA for the development of the successor of the CYBER 205. Until 1987 they will develop the ETA^{10} with 5 nsec cycle time, up to 8 processors, each one with two pipes. Each processor will have a local memory and there will be a very large shared memory, the size depending on the available technology. The maximal hardware vector length of 65535 will be retained because of the compatibility to the CYBER 205. The peak performance for 64 bit arithmetic will be 6.4 GFLOPS. The operating system presently is planned to be VSOS and UNIX.