1985

Using Supercomputers Today and Tomorrow

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
85-529

USING SUPERCOMPUTERS TODAY AND TOMORROW

John R. Rice

CSD-TR-529
August 1985

To appear in the Proceedings of Third
US Army Conference on Applied Mathematics
and Computing, May 1985.

USING SUPERCOMPUTERS TODAY AND TOMORROW

John R. Rice
Computer Science Department
Purdue University
West Lafayette, Indiana 47907
CSD-TR-529
August 15, 1985

ABSTRACT. The past and future growth of supercomputer power is summarized along with the changes in modes of accessing and using supercomputers. Three particular applications are considered from 1983, 1985 and 1995 (hypothetical). It is argued that the software and peripheral support for supercomputers has fallen far behind the increase in power. Solutions to the access and software support are discussed; it is concluded that the access problem is both difficult and very expensive to solve while the software support problem is difficult and only moderately expensive to solve.

I. SUPERCOMPUTER POWER. First, we briefly review the growth in supercomputer power. The peak performance grew slightly less than exponentially over the 1965 to 1980 period. This growth masks the fact that the typical scientist and engineer outside a few weapons laboratory experienced very little growth in the power of the computers available to them. The prices came down but the power did not grow dramatically. Supercomputer power growth has accelerated since 1980 and even more acceleration is forecast for the next 10 years. This growth is summarized in Table 1.

Table 1. Some trends in scientific supercomputing

Year/Machine	*Speed*	*Speed Increase*	
		10-year	*20-year*
1966/CDC6600	1 MFLOPS	–	–
1975/CDC7600	4 MFLOPS	4	–
1980/Cray 1	10 MFLOPS	5	–
1985/Cyber 205	100 MFLOPS	25	100
1990/–	2 GFLOPS	200	1000
1995/–	200 GFLOPS	2000	50,000

The projected 1995 machine has 1000 processors with a 2 nanosecond cycle time.

During the period 1965-1985 there has been a significant change in access to computers. Batch processing has been replaced by some sort of terminal access. The types

of access vary considerably and are listed below in what we think is decreasing frequency:

1. Terminal attached to front end machine connected to network connected to supercomputer
2. Terminal attached to front end machine connected to supercomputer
3. Workstation attached to network attached to front end machine attached to network attached to supercomputer
4. Terminal attached to supercomputer

There are other types of access, but the key point here is that most access is via one or more layers of intermediate machines and networks.

One of the barriers in effective use of supercomputers is the disparate speeds within the intermediate machines and networks. Table 2 presents data on the transfer rates of these facilities.

Table 2. Peak and effective transfer rates of various facilities measured in bits per second (K = 1000, M = 1,000,000)

Facility	*Peak Rate*	*Effective Rate*
Telephone	300	300
2400 baud line	2400	2400
9600 baud line	9600	9600
ARPANET	57K	20K
Bus on VAX 11/780	1M	160K
Ethernet	10M	1.5M
CDC LCN	50M	3M
Cyber 205 channel	200M	100M

It is easy to see that current supercomputers produce results at rates that completely swamp more the capacity of most user's access facilities.

Figure 1 shows the current configuration of the Cyber 205 facility at Purdue University. Most users access the Cyber 205 through the CDC6000 systems or by long haul networks attached to a VAX 11/780.

While progress in access to supercomputers has been significant (yet modest compared to the progress in supercomputer speed), the progress in programming has been uneven and even in reverse for some areas. Editors, on-line file systems, program update systems, libraries, etc. have greatly improved the environment for writing programs. Programming itself has gone downhill. In 1966 Fortran IV was well established. In 1986 supercomputer users can use Fortran 77 (a small improvement), but, if you want to get real supercomputer speeds, you must use machine specific Fortan statements, tricks and generally be rather knowledgeable about the whole machine organization. I believe that the programming task itself is perhaps 40% as efficient on the current Cray and CDC

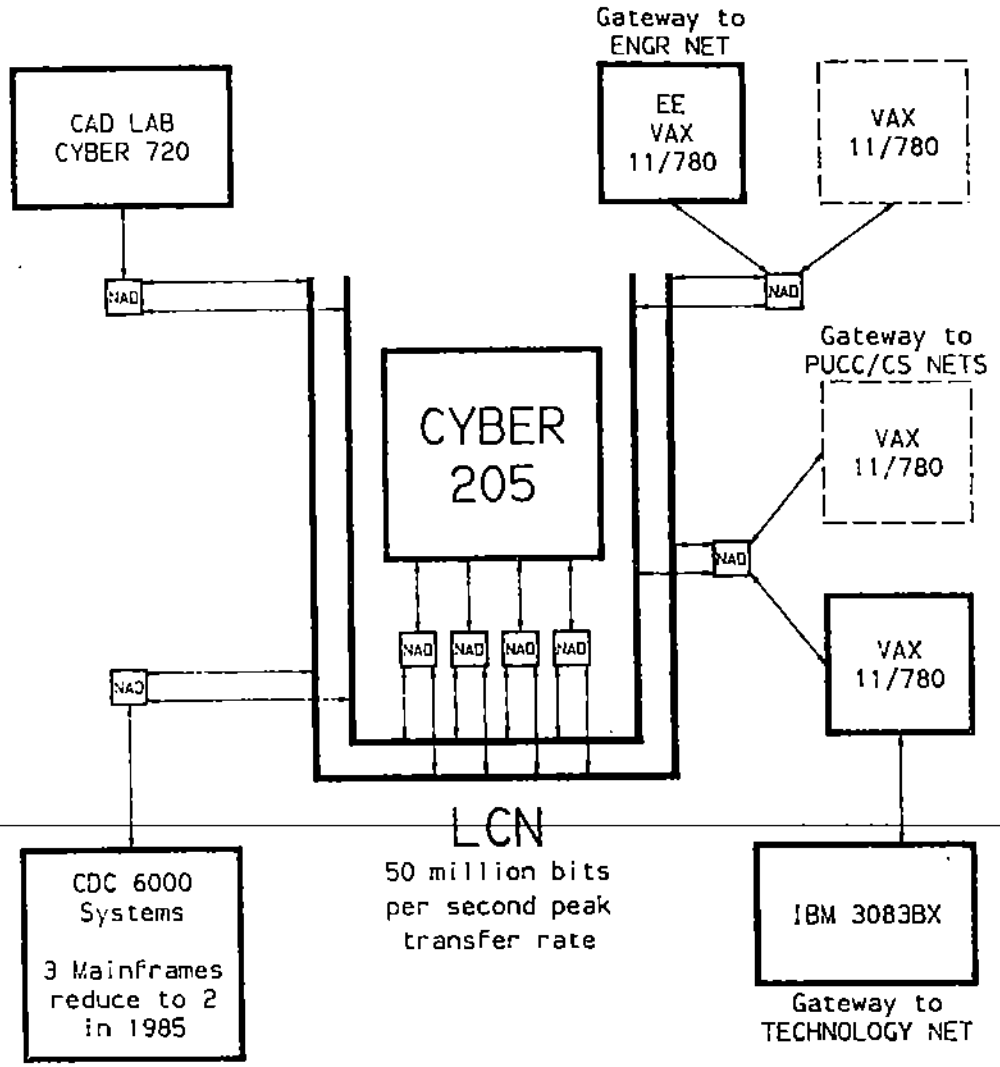

Figure 1. Configuration of the Purdue high speed file transfer network to support the Cyber 205 supercomputer.

supercomputers as it was in 1965 (again, this is for achieving something close to the potential of the machines). Automatic vectorizers are very worthwhile, but they also fall far short of providing the vectorization possible.

I illustrate this development in programming with two randomly selected examples. In [DoEi84] we see the subprogram

```
      SUBROUTINE SMXPY (N1, Y, N2, LDM, X, M)
      REAL Y(*), X(*), M(LDM, *)
      DO 20 J = 1, N2
        DO 10 I = 1, N1
          Y(I) = Y(I) + X(J) * M(I,J)
10     CONTINUE
20     CONTINUE
      RETURN
      END
```

and learn that Cray users should learn to write the inner loop as

```
      DO 10 I = 1, N1
        Y(I) = ((((Y(I) + X(J -3) * M(I, J - 3)) + X(J - 2) * M(I, J -2))
     $             + X(J -1) *M(I, J -1)) + X(J) * M(I,J)
10     CONTINUE.
```

On a Cyber 205 the simple computation

```
FORALL (I=1:NTDIM, J=1:NSDIM) SCORES(I,J) = 60. + 40.*SIN(I*J*63.21)
```

should be programmed as

```
      SEQ(1;NSDIM) = Q8VINTL(1,1;SEQ(1;NSDIM))
      DO 100 I = 1,NTDIM
        ARG(1;NSDIM) = I*63.21*SEQ(1;NSDIM)
        SEQ0(1;NSDIM) = VSIN(ARG(1;NSDIM);SEQ0(1;NSDIM))
        SCORES(I,1;NSDIM) = 60. + 40.*SEQ0(1;NSDIM)
100    CONTINUE
```

in order to achieve high speed execution.

We conclude that the software and peripheral support for supercomputers has fallen far behind the increase in supercomputer computational power.

II. THE SIZE OF SUPERCOMPUTER ANSWERS. Everyone in the supercomputer area and most outside it visualize that supercomputer applications use enormous amounts of computation, millions and billions and trillions of arithmetic steps. Much less widely known is that the answers produced in a typical supercomputer applications are also huge. By answer, we mean the information the user needs in order to understand the computed solution; we do not mean the total set of numerical results computed, which is usually very much larger. We illustrate this with three sample applications, two real ones from 1983 and 1985 and one hypothetical one from 1995.

1983 APPLICATION: *The high speed impact of two steel cubes into a block of aluminum.* This computation was performed at Los Alamos [Los83] on a Cray 1 and used 30 mintues to cover 2.5 microseconds of real time. The problem is eight-dimensional with 3 space variables, time and 4 dependent variable (temperature,

pressure, density of steel and density of aluminum).

Thirty minutes of Cray time represents about 150 billion instructions (12 nanosecond cycle time) including about 18 billion arithmetic operations (10 MFLOPS). The answer can be represented by data on a 100 by 80 by 80 special grid for 150 time steps; each of the 96 million grid points has 4 values (64 bits long). Thus only 400 million numbers represent the result of 18 billion computed numbers, or the answer requires only 4.5% of the numbers computed. Some nice color plots are given in [Los83] and illustrate the effectiveness of this medium for presenting information about computed results. Note that the answer is 3 gigabytes in size which is close to the size of the entire disk memory space on many large scale computer systems.

1985 APPLICATION: *Accretion of material into a black hole (2D model).* This computation [Sm85] shows the evolution of a black hole over a period of millions of years. It assumes axial symmetry to reduce the problem to a feasible size. The answer is 1.25 billion numbers (10 gigabytes). The author discusses how to view the results using color movies. He notes that his computation only provides moderate resolution in time and space and that a good quality movie would require considerably more computation and produce a considerably larger answer.

In this same issue of Science magazine there is a discussion by Joy and Gage [JoGa85] which analyzes the information flow required to produce color movies. Modest resolution, slow motion requires 250 Kbytes/sec while high resolution, normal motion requires about 20 Mbytes/sec. The author argues that color movies are the only way to really assimilate the results of many supercomputer computations.

I estimate that a 3D black hole model giving comparable accuracy would have about 1.5 terabytes in the answer. This would produce a 100 hour movie with normal motion and modest resolution.

1995 APPLICATION: *Tank battle simulation.* In this hypothetical application we assume there are six tanks and the study focuses on the weapons system, the targeting system, the armor and the defensive systems. Thus, intensive, detailed computational analysis is made of a special event such as a shell hit, laser strike or mine explosion. In one of these special events, the physics is followed at the level of the shell explosion, shell case fragmentation and attempted penetration of the armor by blast pressure and heat. Other aspects such as mechanics of the tanks or terrain is simulated at a much coarser level.

A summary of the computation is as follows:

Independent variables:	3 space, time, input of 6 tank drivers, input of 6 tank gunners
Dependent variables:	tank positions
	state of all weapons systems
	state of all defensive systems
	effects of all special events

Computation:	1 hour 2 mega-giga instructions (2 nanosecond cycle, 1000 processors) 700 MFLOPS (200 teraFLOPS machine)
Answer:	A. General Scene: 200 by 200 by 50 grid B. One tank geometry: 100 by 100 by 25 C. One tank weapon system: 10,000 variables D. One tank defensive system: 10,000 variables E. Tank mechanics: 2000 variables F. High level battle scene: 5000 variables, 5000 time steps G. Special Events: 200 events with 200 x 100 x 100 x 200 grid

The size of the answer is then (in megawords)

$$
\begin{aligned}
& A + 6B + 6C + 6D + 6E + F + G \\
= \; & 2 + 6(.25 + .01 + .01 + .002) + 25 + 200 * 400 \\
\sim \; & 1{,}000{,}000
\end{aligned}
$$

Thus the size of this answer is about 8 terabytes. This answer could be shown, in full, as a color movie with normal motion and high resolution that lasts about 100-120 hours. We visualize that the study of this application would involve several people viewing different parts of the answer over a period of time.

We now pose the question: *Suppose the answer has been computed and resides in the supercomputer system, how long will it take to move the answer to the user's location?* We use the effective transfer rates from Table 2 plus the size of answers to produce the results of Table 3. It is obvious from Table 3 that systems which separate the user from the supercomputer by 2 ethernets and a VAX are totally unable to provide reasonable service for many supercomputer applications. Few will put up with waiting a week to see the results of a 30 minute computation. And once he gets the answer "locally" the user neither has a place to put it nor adequate means to view it.

III. SUPERENVIRONMENTS.

We draw three conclusions from the above material:

1. Today's peripherals/workstations/networks are grossly inadequate even for todays supercomputations.
2. Today's programming environments/languages for supercomputers are grossly inadequate, even antiquated.
3. Raw computing power will increase dramatically in the next decade.

The peripheral/workstation/network problem is not easily solved. Fiber optics networks provide a great deal of capacity for networks, but are not yet very cheap. Moderately priced terabit memory systems are not now on the horizon. Workstations with high quality color graphics and movie capabilities are quite expensive and probably

Table 3. Times to transfer the answers of the three applications using various facilities.

		Application	
Facility	*1983*	*1985*	*1995*
Telephone	3 years	9 years	6 millennia
9600 baud line	1 month	3 months	2 centuries
ARPANET	2 weeks	7 weeks	1 century
VAX 11/780 bus	2 days	6 days	7 years
Ethernet	5 hours	15 hours	16 months
Cyber 205 channel	4 min	13 min	1 week
Run time	30 min	1 hour	1 hour

will remain so for some time. For the next decade it may well be that large organizations will have a few superworkstations which are shared by a large community of users.

The programming environment/language problem has many technical difficulties to be overcome, but the initial problem is simply lack of effort. The software support for supercomputers is very meager. We have senior scientists and engineers using facilities that would be instantly rejected by travel agents, junior high math students, secretaries and the general public. It is incredible to see one of the nation's scarest human resources wasted due to the lack of a modest investment (compared to the other aspects of supercomputing) in software support.

One key software area is very high level languages appropriate for scientific computations. Figures 2-4 show three examples of the kind of things we should expect. We do not discuss the ELLPACK [RiBo85], DEQSOL [Ume83] or PROTRAN [AiRi83] systems in any detail but do note they have the following characteristics:

1. They dramatically improve programming productivity.
2. They were implemented with moderate efforts (2-4 man years).
3. They improve execution time efficiency.

Each of these languages has short-commings that one would not expect in production quality systems for supercomputers, yet they represent a great advance over the software currently supplied with supercomputers.

The other key software area is how to map computations onto complex supercomputer architectures so as to produce high efficiency execution. This is a challenging technical problem where many approaches are being actively pursued. However, there is still little indication that the existing and future techniques will be embodied into good user-oriented tools or systems.

We close with Figure 5 which shows the schematic of a superworkstation which is appropriate for supercomputers. It will cost 10-50 times as much as the current good workstation. It needs supersoftware that will also cost 4-50 times as much as current software for supercomputers. The superworkstation and supersoftware are equally important, but the total investment for the software will be an order of magnitude less. Then one might have the superenvironment to take full advantage of the supercomputer power that will appear in the next decade.

IV. ACKNOWLEDGEMENTS. This work was supported in part by the United State Army Research Office, DAAG29-83-K-0026 and the United States Air Force Office of Scientific Research, AFORS-84-0385.

V. REFERENCES.

[AiRi83] T.J. Aird and J.R. Rice, PROTRAN: Problem solving software. Adv. Engin. Software, 5 (1983) 202-206.

[JoGa85] W. Joy and J. Gage, Workstations in science. Science, **228** (1985) 467-470.

[Los83] Computers and Computing. Los Alamos Science, Winter/Spring (1983), 140-141.

[RiBo85] J.R. Rice and R.F. Boisvert, Solving Elliptic Problems using *ELLPACK*. Springer-Verlag, New York (1985).

[Sm85] L.L. Smarr, An approach to complexity: Numerical computations. Science, **228** (1985) 403-408.

[Ume83] Y. Umetani, M. Tsuji, K. Iwasawa and H. Hirayama, DEQSOL: A numerical simulation language for vector/parallel processors. Hitachi report (1983).

```
*                THE PLATEAU PROBLEM
EQUATION. (1.+UY(X,Y)**2) UXX  +  (1.+UX(X,Y)**2) UYY          &
                - 2.*UX(X,Y)*UY(X,Y)  UXY                      &
        + 2.*(UX(X,Y)*UYY(X,Y) - UY(X,Y)*UXY(X,Y)) UX          &
        + 2.*(UY(X,Y)*UXX(X,Y) - UX(X,Y)*UXY(X,Y)) UY          &
    =     2.*(UX(X,Y)*UYY(X,Y) - UY(X,Y)*UXY(X,Y))*UX(X,Y)    &
        + 2.*(UY(X,Y)*UXX(X,Y) - UX(X,Y)*UXY(X,Y))*UY(X,Y)
*
BOUNDARY. U = BOUND(X,Y) ON Y = 0.0  $ U = BOUND(X,Y) ON Y = 1.0
          U = BOUND(X,Y) ON X = 1.0  $ U = BOUND(X,Y) ON X = 0.0
*
GRID.             5 X POINTS  $  5 Y POINTS
TRIPLE.           SET ( U = ZERO )
fORTRAN.
                  DO 100 IT = 1,5
DISCRETIZATION.       HERMITE COLLOCATION
SOLUTION.             BAND GE
FORTRAN.
  100             CONTINUE
OUTPUT.           PLOT(U)
SUBPROGRAMS.
      FUNCTION BOUND(X,Y)
      BOUND = SIN(X+AMAX1(.66,.1+Y**2))*EXP(X-Y)
      RETURN
      END
END.
```

Figure 2. An ELLPACK program that solves the Platean problem (the soap film problem) using Newton iteration combined with Hermite-cubic collocation for the linearized problem.

```
      PARAMETER ( N = 16 )
      REAL MATRIX HILBERT(N,N), X(N,4), B(N,4), RESID(N,4)
      REAL VECTOR RNORM(4)
C                 CREATE HILBERT MATRIX
      ASSIGN HILBERT(I,J) = 1/(I+J-1.)
C                 DEFINE HILBERT MATRIX, FIRST 3 RIGHT SIDES
      ASSIGN   B(I,1) = 0.0
               B(I,2) = 1.0
               B(I,3) = 1.0 + .01*SIN(100.*I)
      B(1,1) = 1.0
C                 COMPUTE 4TH SIDE TO MAKE SOLUTION =1.
      DO 20 I = 1,N
   20      SUM HILBERT(I,J); FOR (J=1,N); IS B(I,4)
C                  SOLVE THE 4 SYSTEMS AND COMPUTE NORM OF RESIDUALS
      LINSYS HILBERT*X = B ; HIGHACCURACY ; SAVE HILBERT
      PRINT B,X
      ASSIGN   RESID = HILBERT*X - B
               RNORM    = RESID'*RESID
               RNORM(K) = SQRT(RNORM(K))
      PRINT RNORM
      END
```

Figure 3. A PROTRAN program that creates a Hilbert matrix and four right sides, solves these linear systems and prints the least squares residual for each system.

```
VAR       TT;
DOM       x = [0:1],
          y = [0:1.2]  ;
TDOM      t = [0:1]
MESH        x = [0.0:1.0:0.2]  ,
            y = [0.0:1.2:0.2]  ,
            t = [0.0:1.0:0.02]  ;
CONST       A = 0.62  ;
REGION      R = [*,*]  ,
            L = [0,*0  ,
            R1 = [1,*]  ,
            D = [*,0]  ,
            U = [*,1.2]  ;
EQU             dt[TT] = A*[lapl [TT] ]  ;
INIT            TT = 100  AT R  ;
BOUND           TT = 200  AT D,
                TT = 200  AT U,
                dx[TT] = 0  AT L+R1;
SCHEME  ;
   ITER          NT UNTIL   NT GT 4  ;
                 TT<+1>=TT+DLT*A*lapl [TT]
   PRINT        TT   AT R ;
   DISP         TT   AT R ;
   END ITER  ;
END SCHEME  ;
END  ;
```

Figure 4. A DEQSOL program that solves a time dependent partial differential equation. The solution obtained this way ran three times as fast as the same method implemented in ordinary Fortran and then hand optimized for vector speeds.

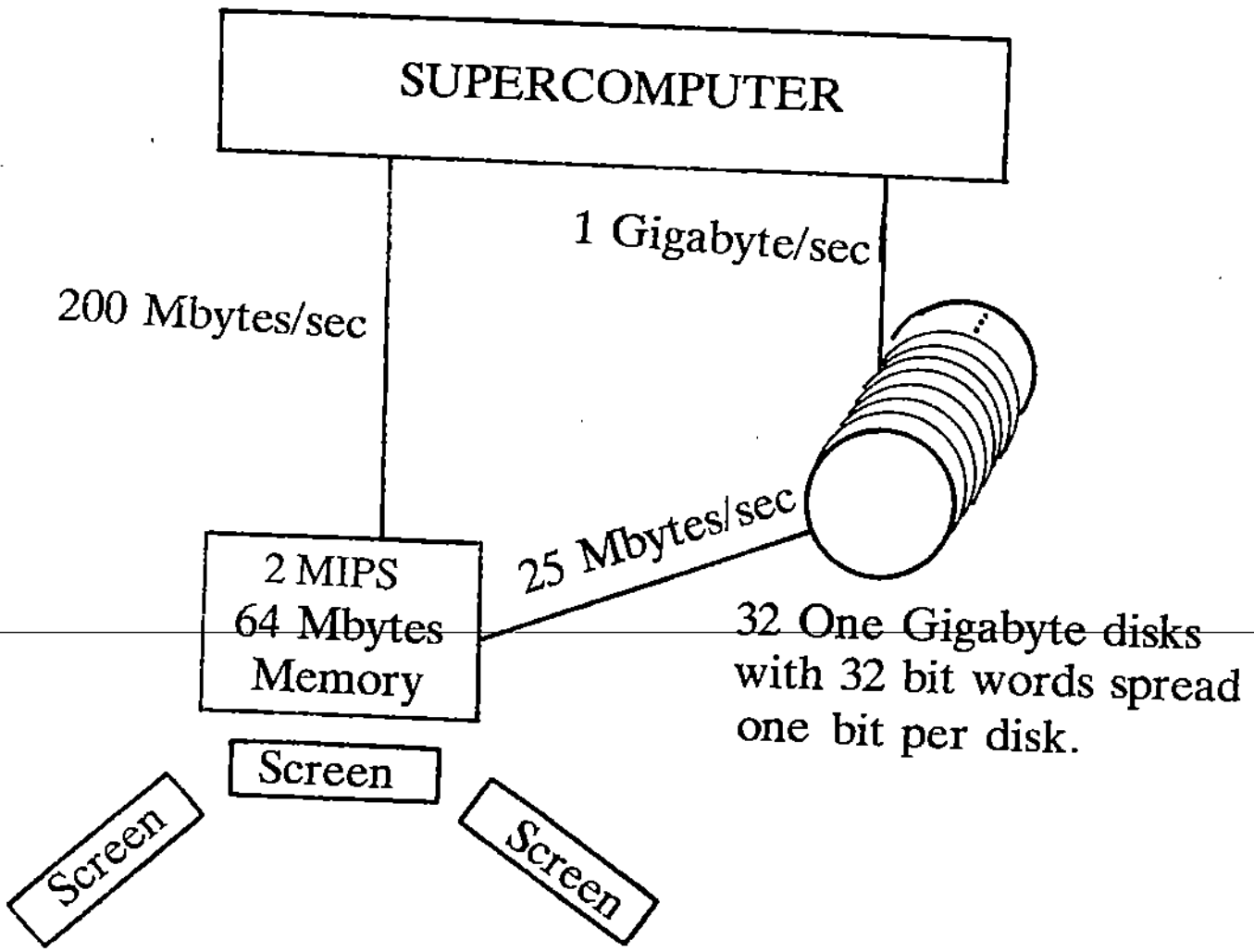

Figure 5. Schematic diagram of a superworkstation appropriate for the supercomputers of the 1990's.