RC9000 Product Management
April 18 1990

R C 9 0 0 0

Performance Benchmarks

April 90

Vedlagt foreløbig sammenfatning af:

1. AIM II tests udført på RC's udstyr samt på MIPS M800.

2. AIM III tests udført på RC9000 og RC990.

3. AIM III testrapporter på andre leverandørers udstyr.

4. Informix Turbo 1.1 målinger.

Kommentarer:

ad1. AIM-II testen er en single-user, single function test, hvor hver test afvikles en ad gangen.
CPU-performance på RC9000/16Mhz er som forventet. I/O performance er i single user mode gns. 80% af RC990 og 43% af M800. Tests på en foreløbig udgave af TX2.2 indikerer en forbedring af I/O på 1,5-2,5 gange, uden at det sikre og robuste filsystem sættes over styr. Single-user I/O performance vil dermed være på højde med tilsvarende MIPS-systemers performance. Systemkald vil muligvis blive forbedret i TX2.2.
For R3000/30Mhz har vi beregnet en forbedring på CPU hastighed på 1,8 gang.

ad2. AIM-III testen er en multi-user load test, som rater et systems evne til håndtering af mange brugere/mange samtidige jobs. Resultaterne er normaliseret i forhold til VAX 11/780, som er ratet som en 12 bruger maskine. Ved alle tests er AIM's standard testprogram-mix anvendt, hvorved resultaterne kan sammenlignes på tværs af produkterne.

Den målte user rating på RC9000 og RC990 er noget usikker, idet testen er megtet følsom over for konfigurerings-og systemparametre. De viste kurver skal derfor tages med et gran salt, vi har ikke haft tid til at gennemføre mere langvarige og kontrollerede målinger. Et helt gennemløb fra 10 til 100 brugere tager et døgn.

I AIM-III testen på RC9000 er real-time og CPU-time meget nær identiske, hvilket viser at I/O systemet ikke er flaskehals (RPU'en kan tage fra ved maximal testload).

Af kurverne ses at vi med RC9000MR ligger i den nedre del af 'konkurrentspektret'(70-80brugere) og at vi med RC9000 Model 35 vil ligge i den øvre del af konkurrentspekret (mere end 110 brugere).

Med den kommende MP-version af PU'en, forventer vi at opnå relative performanceforbedringer der svarer til Pyramids og Sequents tilsvarende tal.

AIM-III testen kan konfigureres til at teste et systems evne som office-maskine. Sættes profilen til 70% wordprocessing og 30% spreadsheet viser testen at RC9000MR kan trække 70 samtidige brugere.

AIM-III testen kan afvikles på et multi-PU system (systemkald Exec udskiftes med systemkald Run). Dette betyder at vi kan rate et FT-og multi-PU system. Test vil blive gennemført i den nærmeste tid.

ad3. AIM performance rapporter på "konkurrentudstyr".
Der er kun valgt 'single-CPU' systemer, dog med undtagelse af Sequent S27, som har to 80386 CPU'er.
Den nuværende RC9000MR performer bedre end en Pyramid 9815 og RC9000 model 35 forventes at performe bedre end HP9000/835. IBM RISCSystem6000 Model 530 slår dog alle.

ad4. Jvfr. de vedlagte resultater fra TP1-målingerne kan Informix-Turbo 1.1 på RC9000MR yde op til 16 TP1. I kommentarerne til Turbo 1.1 fremgår det, at implementationen på visse punkter er uheldig med hensyn til optimal performance. Dette er rettet i Informix-Online. En tidlig version af Online er anvendt i "The Turbo Trials Project", hvor der med Sequent S27 (2 processorer) er målt performance på 30 TP1.
Sjovt nok har Informix ikke publiceret TP1 tal for Turbo 1.1 (mig bekendt), men fra MIPS ved vi, at den tilsvarende MIPS maskine M1500 yder en performance på 17 TP1.
I DSR's 4GL applikationsmix er det stikprøvemæssigt registreret, at 35 samtidige brugere belaster en RC9000MR 77 % (CPU-tid).
Med samme applikations-og loadmix vil RC9000 Model 35 formodentligt kunne understøtte op til 80 samtidige brugere.
Det er ikke muligt at vurdere, hvilken forbedring Informix-Online vil give.

A I M II T E S T

AIM II Tests RC9000, M800, RC990/486 HLJ

Note: For RCI Internal use only.

Test			TX 2.1 R2000/16Mhz Abs.	Rel.	RISCos R2000/8Mhz Rel.	RC990 486/25mhz Rel.	TX 2.2 R2000/16Mhz Rel.	TX 2.2 R3000/30Mhz Rel.
Arithmetic instruction times	add	short	77ns	1	2,2	4,7	1	
		long	73ns	1	2,2	1,4	1	
		float	259ns	1	340	3,9	1	
		double	187ns	1	294	3,9	1	
	mult	short	918ns	1	2,2	1,1	1	
		long	909ns	1	2,2	1,1	1	
		float	341ns	1	319	2,9	1	
		double	304ns	1	233	3,1	1	
	div	short	2000ns	1	2,2	1,5	1	
		long	2000ns	1	3,0	1,5	1	
		float	1000ns	1	100	5,0	1	
		double	1000ns	1	69	4,0	1	
Memory loop access times	char	read	141ns	1	2,2	2,3	1	
		write	556ns	1	2,2	0,7	1	
		copy	527ns	1	2,2	1,3	1	
	short	read	71ns	1	2,2	3,2	1	
		write	302ns	1	1,9	0,7	1	
		copy	269ns	1	2,0	1,5	1	
	long	read	37ns	1	2,2	2,2	1	
		write	134ns	1	2,1	0,7	1	
		copy	133ns	1	2,0	1,3	1	
Input/ output rates (kbytes/s)		read	297	1	4,6	1,1	? read-ahead implementeres	
		write	158	1	2,5	1,5	2,9	
		copy	81	1	4,1	1,6	? read-ahead implementeres	
		pipe	634	1	1,6	1,6	1,4	
	RAM	1-byte	1897	1	0,5	0,8	1,0	
	RAM	4-byte	7493	1	0,5	0,8	1,0	
Array subscript ref. (ns/ref)		short	642ns	1	1,6	0,4	1,0	
		long	224ns	1	2,2	0,8	1,0	
Function ref. (ns/ref)		0-par.	262ns	1	2,2	3,8	1,0	
		1-par.	457ns	1	2,2	4,4	1,0	
		2-par.	827ns	1	2,4	2,4	1,0	
Process forks (antal/s)			35	1	2,3	2,8	?	
Getpid (kcalls/s)			33	1	0,5	0,6	?	
Sbrk(0) (kcalls/s)			17	1	0,6	31,2	?	
Create/close (pairs/s)			163	1	3,8	0,2	?	
Umask(0) (kcalls/s)			26	1	0,6	0,7	?	

```
What is  this machine's name: qa990 386

What is its price in dollars: 100000

Testing Started
TEST00 qa990 386                    100000.
TEST01 getpid() calls                 8396:    8 kcalls/sec or   119 microsec/call
TEST02 sbrk(0) calls                195123:  195 kcalls/sec or     5 microsec/call
TEST03 creat-close calls                36:   36 pairs/sec or 27778 microsec/pair
TEST33 umask(0) calls                 7348:    7 kcalls/sec or   136 microsec/call
TEST04 process forks                    61:   61 forks/sec or 16393 microsec/fork
TEST06 disk write                   185969:  186 kbytes/sec or     5 microsec/byte
TEST07 disk read                    345409:  345 kbytes/sec or     3 microsec/byte
TEST08 disk copy                    127401:  127 kbytes/sec or     8 microsec/byte
TEST09 pipe copy                    668614:  669 kbytes/sec or     1 microsec/byte
TEST10 add LONG                    4621593: 4622 kadds/sec or   216 nanosec/add
TEST11 add SHORT                   2432128: 2432 kadds/sec or   411 nanosec/add
TEST12 add FLOAT                    145022:  145 kadds/sec or     7 microsec/add
TEST34 add DOUBLE                   330364:  330 kadds/sec or     3 microsec/add
TEST13 array ref short[short]      1957378: 1957 krefs/sec or   511 nanosec/ref
TEST14 array ref long[long]        2174867: 2175 krefs/sec or   460 nanosec/ref
TEST15 call funct()                 301176:  301 kcalls/sec or     3 microsec/call
TEST16 call funct(int)              213537: .214 kcalls/sec or     5 microsec/call
TEST17 call funct(int,int)          169179:  169 kcalls/sec or     6 microsec/call
TEST18 ram read SHORT              1891656: 1892 kbytes/sec or   529 nanosec/byte
TEST19 ram read LONG               4376927: 4377 kbytes/sec or   228 nanosec/byte
TEST20 ram read CHAR               1238909: 1239 kbytes/sec or   807 nanosec/byte
TEST21 ram write SHORT             2048000: 2048 kbytes/sec or   488 nanosec/byte
TEST22 ram write LONG              4347580: 4348 kbytes/sec or   230 nanosec/byte
TEST23 ram write CHAR              1181538: 1182 kbytes/sec or   846 nanosec/byte
TEST24 ram copy SHORT              1026926: 1027 kbytes/sec or   974 nanosec/byte
TEST25 ram copy LONG               2174978: 2175 kbytes/sec or   460 nanosec/byte
TEST26 ram copy CHAR                579622:  580 kbytes/sec or     2 microsec/byte
TEST27 multiply SHORT               839696:  840 kmults/sec or     1 microsec/mult
TEST28 multiply LONG                990808:  991 kmults/sec or     1 microsec/mult
TEST29 multiply FLOAT               155091:  155 kmults/sec or     6 microsec/mult
TEST35 multiply DOUBLE              267504:  268 kmults/sec or     4 microsec/mult
TEST30 divide SHORT                 324531:  325 kdivs/sec or    3 microsec/div
TEST31 divide LONG                  310277:  310 kdivs/sec or    3 microsec/div
TEST32 divide FLOAT                  88797:   89 kdivs/sec or   11 microsec/div
TEST37 divide DOUBLE                135024:  135 kdivs/sec or    7 microsec/div
Testing Completed.
```

ARITHMETIC INSTRUCTION TIMES (microseconds per op)

	short	long	float	double
+ add	411ns	216ns	7	3
* multiply	1	1	6	4
/ divide	3	3	11	7

MEMORY LOOP ACCESS TIMES (nanoseconds per byte)

	read	write	copy
CHAR type	807ns	846ns	2
SHORT type	529ns	488ns	974ns
LONG type	229ns	230ns	460ns

INPUT/OUTPUT RATES (bytes/sec)

	read	write	copy
DISK	345k	186k	127k
PIPE			669k
TTY 1		0	
TTY 1+2		0	
RAM 1-byte			580k
RAM 4-byte			2175k

```
ARRAY SUBSCRIPT REFERENCES  (nanoseconds)
   short[]          long[]
    511 ns         460 ns

FUNCTION REFERENCES  (microseconds/ref)
 0-parameters     1-parameter     2-parameters
   funct()          funct(i)        funct(i,i)
      3                5                 6

PROCESS FORKS
 (4215k bytes)
61 per second

SYSTEM KERNEL CALLS     (calls-per-second and microseconds per call)
getpid() calls:       8 kcalls/sec or     119 microseconds/call
sbrk(0)  calls.       195 kcalls/sec or       5 microseconds/call
create/close calls:   36 pairs/sec or  27776 microseconds/pair
umask(0) calls:   7 kcalls/sec or    136 microseconds/call
```

```
What is  this machine's name. bakken

What is its price in dollars: 100000

Testing Started
TEST00 bakken                  100000
TEST01 getpid() calls           17311:    17 kcalls/sec or    58 microsec/call
TEST02 sbrk(0) calls            10941:    11 kcalls/sec or    91 microsec/call
TEST03 creat-close calls          625    625 pairs/sec or 1600 microsec/pair
TEST33 umask(0) calls           17023     17 kcalls/sec or    59 microsec/call
TEST04 process forks               80     80 forks/sec or 12500 microsec/fork
TEST06 disk write              390701    391 kbytes/sec or     3 microsec/byte
TEST07 disk read              1363098   1363 kbytes/sec or   734 nanosec/byte
TEST08 disk copy               334602    335 kbytes/sec or     3 microsec/byte
TEST09 pipe copy              1015077:  1015 kbytes/sec or   985 nanosec/byte
TEST10 add LONG               6203077:  6203 kadds/sec or   161 nanosec/add
TEST11 add SHORT              5865794:  5866 kadds/sec or   170 nanosec/add
TEST12 add FLOAT                11389:    11 kadds/sec or    88 microsec/add
TEST34 add DOUBLE               18101:    18 kadds/sec or    55 microsec/add
TEST13 array ref short[short]  863472:   863 krefs/sec or     1 microsec/ref
TEST14 array ref long[long]   2003618:  2004 krefs/sec or   499 nanosec/ref
TEST15 call funct()           1738986:  1739 kcalls/sec or   575 nanosec/call
TEST16 call funct(int)         962571:   963 kcalls/sec or     1 microsec/call
TEST17 call funct(int,int)     629859:   630 kcalls/sec or     2 microsec/call
TEST18 ram read SHORT         6277921:  6278 kbytes/sec or   159 nanosec/byte
TEST19 ram read LONG         12406154: 12406 kbytes/sec or    81 nanosec/byte
TEST20 ram read CHAR          3169242:  3169 kbytes/sec or   316 nanosec/byte
TEST21 ram write SHORT        1729754:  1730 kbytes/sec or   578 nanosec/byte
TEST22 ram write LONG         3554991:  3555 kbytes/sec or   281 nanosec/byte
TEST23 ram write CHAR          886206:   886 kbytes/sec or     1 microsec/byte
TEST24 ram copy SHORT         1854113:  1854 kbytes/sec or   539 nanosec/byte
TEST25 ram copy LONG          3697778:  3698 kbytes/sec or   270 nanosec/byte
TEST26 ram copy CHAR           847448:   847 kbytes/sec or     1 microsec/byte
TEST27 multiply SHORT          490193:   490 kmults/sec or     2 microsec/mult
TEST28 multiply LONG           495404:   495 kmults/sec or     2 microsec/mult
TEST29 multiply FLOAT            9159:     9 kmults/sec or   109 microsec/mult
TEST35 multiply DOUBLE          14148:    14 kmults/sec or    71 microsec/mult
TEST30 divide SHORT            194536:   195 kdivs/sec or     5 microsec/div
TEST31 divide LONG             181016:   181 kdivs/sec or     6 microsec/div
TEST32 divide FLOAT              9625:    10 kdivs/sec or   104 microsec/div
TEST37 divide DOUBLE            14430:    14 kdivs/sec or    69 microsec/div
Testing Completed.
```

ARITHMETIC INSTRUCTION TIMES (microseconds per op)

	short	long	float	double
+ add	170ns	161ns	88	55
* multiply	2	2	109	71
/ divide	5	6	104	69

MEMORY LOOP ACCESS TIMES (nanoseconds per byte)

	read	write	copy
CHAR type	316ns	1	1
SHORT type	159ns	578ns	539ns
LONG type	81ns	281ns	270ns

INPUT/OUTPUT RATES (bytes/sec)

	read	write	copy
DISK	1363k	391k	335k
PIPE			1015k
TTY 1		0	
TTY 1+2		0	
RAM 1-byte			847k
RAM 4-byte			3698k

```
ARRAY SUBSCRIPT REFERENCES   (microseconds)
  short[]          long[]
     1             499 ns

FUNCTION REFERENCES  (nanoseconds/ref)
 0-parameters     1-parameter     2-parameters
  funct()          funct(i)        funct(i,i)
    575               1                2

PROCESS FORKS
 (264263k bytes)
80 per second

SYSTEM KERNEL CALLS      (calls-per-second and microseconds per call)
getpid() calls:       17 kcalls/sec or      58 microseconds/call
sbrk(0)  calls:       11 kcalls/sec or      91 microseconds/call
create/close calls:   625 pairs/sec or    1600 microseconds/pair
umask(0) calls    17 kcalls/sec or      59 microseconds/call
```

```
What is  this machine's name: zonker TX2 2

What is its price in dollars: 100000

Testing Started.
TEST00 zonker TX2 2             100000.
TEST01 getpid() calls            33191     33 kcalls/sec or    30 microsec/call
TEST02 sbrk(0) calls             19564:    20 kcalls/sec or    51 microsec/call
TEST03 creat-close calls           161:   161 pairs/sec or 6211 microsec/pair
TEST33 umask(0) calls            30952:    31 kcalls/sec or    32 microsec/call
TEST04 process forks                35:    35 forks/sec or 28571 microsec/fork
TEST06 disk write               462522.   463 kbytes/sec or     2 microsec/byte
TEST07 disk read                313097:   313 kbytes/sec or     3 microsec/byte
TEST08 disk copy                134407:   134 kbytes/sec or     7 microsec/byte
TEST09 pipe copy                903630:   904 kbytes/sec or     1 microsec/byte
TEST10 add LONG               13655812: 13656 kadds/sec or    73 nanosec/add
TEST11 add SHORT              13083900. 13084 kadds/sec or    76 nanosec/add
TEST12 add FLOAT               3874595:  3875 kadds/sec or   258 nanosec/add
TEST34 add DOUBLE              5362660:  5363 kadds/sec or   186 nanosec/add
TEST13 array ref short[short]  1575385:  1575 krefs/sec or   635 nanosec/ref
TEST14 array ref long[long]    4455205:  4455 krefs/sec or   224 nanosec/ref
TEST15 call funct()            3863309:  3863 kcalls/sec or  259 nanosec/call
TEST16 call funct(int)         2155790:  2156 kcalls/sec or  464 nanosec/call
TEST17 call funct(int,int)     1212521:  1213 kcalls/sec or  825 nanosec/call
TEST18 ram read SHORT         14016191: 14016 kbytes/sec or   71 nanosec/byte
TEST19 ram read LONG          27439225: 27439 kbytes/sec or   36 nanosec/byte
TEST20 ram read CHAR           7088776:  7089 kbytes/sec or  141 nanosec/byte
TEST21 ram write SHORT         3735307:  3735 kbytes/sec or  268 nanosec/byte
TEST22 ram write LONG          7540364:  7540 kbytes/sec or  133 nanosec/byte
TEST23 ram write CHAR          1920000:  1920 kbytes/sec or  521 nanosec/byte
TEST24 ram copy SHORT          3737081:  3737 kbytes/sec or  268 nanosec/byte
TEST25 ram copy LONG           7585579:  7586 kbytes/sec or  132 nanosec/byte
TEST26 ram copy CHAR           1832968:  1833 kbytes/sec or  546 nanosec/byte
TEST27 multiply SHORT          1099827:  1100 kmults/sec or  909 nanosec/mult
TEST28 multiply LONG           1097179:  1097 kmults/sec or  911 nanosec/mult
TEST29 multiply FLOAT          2930329:  2930 kmults/sec or  341 nanosec/mult
TEST35 multiply DOUBLE         3300921:  3301 kmults/sec or  303 nanosec/mult
TEST30 divide SHORT             431647:   432 kdivs/sec or     2 microsec/div
TEST31 divide LONG              404264:   404 kdivs/sec or     2 microsec/div
TEST32 divide FLOAT             851332:   851 kdivs/sec or     1 microsec/div
TEST37 divide DOUBLE            869526:   870 kdivs/sec or     1 microsec/div
Testing Completed.

ARITHMETIC INSTRUCTION TIMES  (microseconds per op)

                     short   long   float   double
---------------------------------------------------
+ add                 76ns   73ns   258ns   186ns

* multiply           909ns  911ns   341ns   303ns

/ divide               2      2       1       1


MEMORY LOOP ACCESS TIMES  (nanoseconds per byte)

                      read   write   copy
------------------------------------------
CHAR type            141ns   521ns   546ns

SHORT type            71ns   268ns   268ns

LONG type             36ns   133ns   132ns


INPUT/OUTPUT RATES  (bytes/sec)

                      read   write   copy
------------------------------------------
DISK                  313k   463k    134k

PIPE                                 904k

TTY 1                          0
TTY 1+2                        0

RAM 1-byte                          1833k
RAM 4-byte                          7586k


ARRAY SUBSCRIPT REFERENCES  (nanoseconds)
  short[]         long[]
   635 ns         224 ns

FUNCTION REFERENCES  (nanoseconds/ref)
 0-parameters     1-parameter     2-parameters
  funct()          funct(i)        funct(i,i)
     259              464              825

PROCESS FORKS
 (264281k bytes)
35 per second

SYSTEM KERNEL CALLS     (calls-per-second and microseconds per call)
getpid() calls:        33 kcalls/sec or      30 microseconds/call
sbrk(0)  calls:        20 kcalls/sec or      51 microseconds/call
create/close calls:   161 pairs/sec or     6211 microseconds/pair
umask(0) calls:    31 kcalls/sec or      32 microseconds/call
```

```
What is  this machine s name: qa9000 tx2.1

What is its price in dollars: 100000

Testing Started.
TEST00 qa9000 tx2.1                100000
TEST01 getpid() calls               33279:     33 kcalls/sec or    30 microsec/call
TEST02 sbrk(0) calls                16826:     17 kcalls/sec or    59 microsec/call
TEST03 creat-close calls              163:    163 pairs/sec or 6135 microsec/pair
TEST33 umask(0) calls               25794:     26 kcalls/sec or    39 microsec/call
TEST04 process forks                   35:     35 forks/sec or 28571 microsec/fork
TEST06 disk write                  157733:    158 kbytes/sec or     6 microsec/byte
TEST07 disk read                   297391:    297 kbytes/sec or     3 microsec/byte
TEST08 disk copy                    80743:     81 kbytes/sec or    12 microsec/byte
TEST09 pipe copy                   633676:    634 kbytes/sec or     2 microsec/byte
TEST10 add LONG                  13722277: 13722 kadds/sec or    73 nanosec/add
TEST11 add SHORT                 12940177: 12940 kadds/sec or    77 nanosec/add
TEST12 add FLOAT                  3863063:  3863 kadds/sec or   259 nanosec/add
TEST34 add DOUBLE                 5351003:  5351 kadds/sec or   187 nanosec/add
TEST13 array ref short[short]     1557568:  1558 krefs/sec or   642 nanosec/ref
TEST14 array ref long[long]       4467551:  4468 krefs/sec or   224 nanosec/ref
TEST15 call funct()               3821573:  3822 kcalls/sec or  262 nanosec/call
TEST16 call funct(int)            2187680:  2188 kcalls/sec or  457 nanosec/call
TEST17 call funct(int,int)        1209688:  1210 kcalls/sec or  827 nanosec/call
TEST18 ram read SHORT            14014306: 14014 kbytes/sec or   71 nanosec/byte
TEST19 ram read LONG             27383872: 27384 kbytes/sec or   37 nanosec/byte
TEST20 ram read CHAR              7089231:  7089 kbytes/sec or  141 nanosec/byte
TEST21 ram write SHORT            3311001:  3311 kbytes/sec or  302 nanosec/byte
TEST22 ram write LONG             7466068:  7466 kbytes/sec or  134 nanosec/byte
TEST23 ram write CHAR             1797116:  1797 kbytes/sec or  556 nanosec/byte
TEST24 ram copy SHORT             3715856:  3716 kbytes/sec or  269 nanosec/byte
TEST25 ram copy LONG              7493411:  7493 kbytes/sec or  133 nanosec/byte
TEST26 ram copy CHAR              1897332:  1897 kbytes/sec or  527 nanosec/byte
TEST27 multiply SHORT             1089395:  1089 kmults/sec or  918 nanosec/mult
TEST28 multiply LONG              1099825:  1100 kmults/sec or  909 nanosec/mult
TEST29 multiply FLOAT             2930337:  2930 kmults/sec or  341 nanosec/mult
TEST35 multiply DOUBLE            3292604:  3293 kmults/sec or  304 nanosec/mult
TEST30 divide SHORT                432737:   433 kdivs/sec or     2 microsec/div
TEST31 divide LONG                 404210:   404 kdivs/sec or     2 microsec/div
TEST32 divide FLOAT                849448:   849 kdivs/sec or     1 microsec/div
TEST37 divide DOUBLE               865230:   865 kdivs/sec or     1 microsec/div
Testing Completed

ARITHMETIC INSTRUCTION TIMES  (microseconds per op)

                       short   long    float   double
---------------------------------------------------------
+ add                  77ns    73ns    259ns   187ns

* multiply             918ns   909ns   341ns   304ns

/ divide               2       2       1       1


MEMORY LOOP ACCESS TIMES  (nanoseconds per byte)

                       read    write   copy
--------------------------------------------
CHAR type              141ns   556ns   527ns

SHORT type             71ns    302ns   269ns

LONG type              37ns    134ns   133ns


INPUT/OUTPUT RATES  (bytes/sec)

                       read    write   copy
--------------------------------------------
DISK                   297k    158k    81k

PIPE                                   634k

TTY 1                          0
TTY 1+2                        0

RAM 1-byte                             1897k
RAM 4-byte                             7493k

ARRAY SUBSCRIPT REFERENCES  (nanoseconds)
   short[]          long[]
     642 ns         224 ns

FUNCTION REFERENCES  (nanoseconds/ref)
 0-parameters     1-parameter     2-parameters
   funct()         funct(i)        funct(i,i)
     262             457              827

PROCESS FORKS
 (264277k bytes)
35 per second

SYSTEM KERNEL CALLS       (calls-per-second and microseconds per call)
getpid() calls:         33 kcalls/sec or       30 microseconds/call
sbrk(0)  calls:         17 kcalls/sec or       59 microseconds/call
create/close calls:    163 pairs/sec or      6135 microseconds/pair
umask(0) calls:    26 kcalls/sec or      39 microseconds/call
```

```
is  this machine's name. qa990 486
is its price in dollars. 100000

sting Started
ST00 qa990 486                    100000.
ESTO1 getpid() calls               19689.    20 kcalls/sec or     51 microsec/call
TEST02 sbrk(O) calls              531328.   531 kcalls/sec or      2 microsec/call
TEST03 creat-close calls              35:    35 pairs/sec or 28571 microsec/pair
TEST33 umask(O) calls              18184:    18 kcalls/sec or     55 microsec/call
TEST04 process forks                  97:    97 forks/sec or 10309 microsec/fork
TEST06 disk write                 242704:   243 kbytes/sec or      4 microsec/byte
TEST07 disk read                  335047:   335 kbytes/sec or      3 microsec/byte
TEST08 disk copy                  132883:   133 kbytes/sec or      8 microsec/byte
TEST09 pipe copy                 1033999:  1034 kbytes/sec or    967 nanosec/byte
TEST10 add LONG                 10143397: 10143 kadds/sec or      99 nanosec/add
TEST11 add SHORT                 2759167:  2759 kadds/sec or     362 nanosec/add
TEST12 add FLOAT                  768096:   768 kadds/sec or       1 microsec/add
TEST34 add DOUBLE                1355294:  1355 kadds/sec or     738 nanosec/add
TEST13 array ref short[short]    3746349:  3746 krefs/sec or     267 nanosec/ref
TEST14 array ref long[long]      5729952:  5730 krefs/sec or     175 nanosec/ref
TEST15 call funct()               682667:   683 kcalls/sec or      1 microsec/call
TEST16 call funct(int)            522191:   522 kcalls/sec or      2 microsec/call
TEST17 call funct(int, int)       430739:   431 kcalls/sec or      2 microsec/call
TEST18 ram read SHORT            4441797:  4442 kbytes/sec or    225 nanosec/byte
TEST19 ram read LONG            12172076: 12172 kbytes/sec or     82 nanosec/byte
TEST20 ram read CHAR             3156070:  3156 kbytes/sec or    317 nanosec/byte
TEST21 ram write SHORT           5071221:  5071 kbytes/sec or    197 nanosec/byte
TEST22 ram write LONG           10750818: 10751 kbytes/sec or     93 nanosec/byte
TEST23 ram write CHAR            2759167:  2759 kbytes/sec or    362 nanosec/byte
TEST24 ram copy SHORT            2502554:  2503 kbytes/sec or    400 nanosec/byte
TEST25 ram copy LONG             5753646:  5754 kbytes/sec or    174 nanosec/byte
TEST26 ram copy CHAR             1498537:  1499 kbytes/sec or    667 nanosec/byte
TEST27 multiply SHORT             847448:   847 kmults/sec or      1 microsec/mult
TEST28 multiply LONG             1035582:  1036 kmults/sec or    966 nanosec/mult
TEST29 multiply FLOAT             772830:   773 kmults/sec or      1 microsec/mult
TEST35 multiply DOUBLE           1071589:  1072 kmults/sec or    933 nanosec/mult
TEST30 divide SHORT               306078:   306 kdivs/sec or       3 microsec/div
TEST31 divide LONG                318911:   319 kdivs/sec or       3 microsec/div
TEST32 divide FLOAT               205687:   206 kdivs/sec or       5 microsec/div
TEST37 divide DOUBLE              233126:   233 kdivs/sec or       4 microsec/div
Testing Completed.
```

ARITHMETIC INSTRUCTION TIMES (microseconds per op)

	short	long	float	double
+ add	362ns	99ns	1	738ns
* multiply	1	966ns	1	933ns
/ divide	3	3	5	4

MEMORY LOOP ACCESS TIMES (nanoseconds per byte)

	read	write	copy
CHAR type	317ns	362ns	667ns
SHORT type	225ns	197ns	400ns
LONG type	82ns	93ns	174ns

INPUT OUTPUT RATES (bytes/sec)

	read	write	copy
DISK	335k	243k	133k
PIPE			1034k
TTY 1		0	
TTY 1+2		0	
RAM 1-byte			1499k
RAM 4-byte			5754k

ARRAY SUBSCRIPT REFERENCES (nanoseconds)

short[]	long[]
267 ns	175 ns

FUNCTION REFERENCES (microseconds/ref)

0-parameters funct()	1-parameter funct(i)	2-parameters funct(i, i)
1	2	2

PROCESS FORKS
(4215k bytes)
97 per second

```
SYSTEM KERNEL CALLS     (calls-per-second and microseconds per call)
getpid() calls:          20 kcalls/sec or      51 microseconds/call
sbrk(O)  calls:         531 kcalls/sec or       2 microseconds/call
create/close calls:      35 pairs/sec or    28571 microseconds/pair
umask(O) calls:          18 kcalls/sec or      55 microseconds/call
```

A I M III
R C 9 0 0 0 og R C 9 9 0

Understanding Benchmarks

JIM GEERS
AIM TECHNOLOGY

Increased software portability has greatly expanded the need for benchmarks. Do the benchmarking techniques available today serve the needs of UNIX end users and manufacturers?

The goal of application source code portability in UNIX is approaching reality. Efforts are also underway to establish UNIX binary standards. MSDOS and OS/2 have already achieved binary portability. Soon, the purchase of UNIX computing power for many requirements can be approached as a practical commodity opportunity.

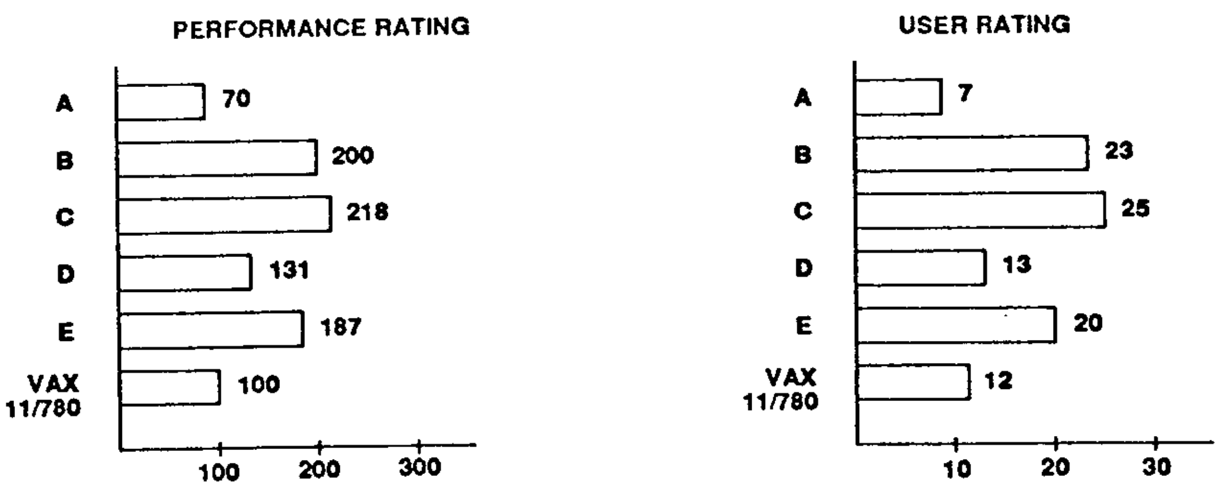
How can both users and manufacturers quickly determine the performance of a wide variety of systems to select those for further evaluation? How can a number of systems be evaluated cost effectively? How can the effect of sophisticated architectural alternatives be quickly analyzed?

Selection and evaluation have become much more complex. Just a few years ago only three or four available computer models might meet a buyer's requirements. Each could be investigated without requiring an inordinate amount of time or cost. In the majority of procurements, a manufacturer competed with the same two or three suppliers and it was much easier to gauge the relative performance capabilities of each system.

Now, depending on the intended use, a buyer is faced with 10, 20 or even 50 models that might potentially meet his needs. Manufacturers now compete with dozens of vendors in a given procurement. A wide variety of sophisticated alternatives that affect the performance of each model are available, including memory & disk caching, graphics enhancement, instruction set alternatives (RISC versus CISC), and memory management schemes. How do each of these alternatives affect overall system performance?

Keep It Simple?

The phrase MIPS (Millions of Instructions Per Second) is frequently given as a quick method of gauging relative system performance. Although there is no universal agreement on how to define MIPS, it certainly is a function of the CPU power and its clock speed. Figure 1 shows just how deceiving this method of measuring system performance can be. The performance rating of five UNIX processors *each using a 68020 with clock speed of 16.7 MHZ* is compared to a VAX 11/780. All are running the identical load.



1049 05

Figure 1. System Performance Variation with Same CPU

The VAX 11/780 is given the nominal rating of 100. The system performance of the five 68020 systems varies from 70% of the VAX to 218% of the VAX, a difference of more than three times the performance!

When the user rating (the measurement of workload throughput) of each of these five systems is compared to the VAX, we see that one system has the capability to handle only 7 users at the tested load level, while another system can handle 25 users executing the identical load. Here the range of performance capability varies by a factor of roughly 3.6.

Much of this difference in system performance derives from the various hardware surrounding the CPU, such as memory and disk subsystems. But software efficiency also contributes heavily to system performance. In Figure 2, five different UNIX ports are run on the same hardware platform. Performance at the high end is more than 50% greater than the low end. The user load capability varies by 46%.

Operating System	Performance Rating	User Rating
A	158	19
B	133	16
C	142	17
D	142	13
E	158	19

All tests conducted on the same hardware system.

Figure 2. System Performance Variation with Different UNIX Operating Systems

Two Benchmarking Points of View

It would appear to be desirable to develop one all encompassing methodology for benchmarking all systems. In reality, this is not a practical goal; in fact, the range of benchmarking tools is expanding, not contracting. Benchmarking technology must take into account the variety of application programs to be run, the quantity and expanding resource requirements of the simultaneous tasks to be performed, and the number of simultaneous users the system will be required to support.

The need for a variety of benchmarking tools can be better understood by examining the user operating environments. Some are single user systems with heavy multitasking demands. Others are multiple user systems where each user performs relatively similar tasks. There are also multiple- user environments where heavy multitasking demands are required.

Manufacturer and end user evaluators are faced with a variety of user environments. They are interested in establishing total system performance and identifying subsystem bottlenecks. They seek to evaluate relative performance among a number of models. Application's strengths and weaknesses need to be understood. This understanding can be used to position various product offerings and, in a commodity market, establish market value.

Face it, designing and marketing a computer is an exercise in compromise. The data obtained from benchmarking is extremely useful for optimizing system design and performance. System parameters can be tuned to achieve a desired system performance. Benchmarking can be used to verify that the design and tuning goals have been achieved.

With all the variations in benchmarking tools, benchmarking can be divided into two general approaches: 1) the view of what the user is doing (User Workload Benchmarking), and 2) the view of what the system is doing (System Benchmarking).

User Workload Benchmarks

Approaching benchmarking from the user simulation perspective is a continuous trade-off between accuracy and cost/time. These can vary greatly depending on the need to simulate the 1) users, 2) programs, and 3) computer system. Figures 3a, 3b, and 3c indicate how these trade-offs are being approached in today's benchmarking.

The challenge in this form of benchmarking is characterizing the workload; that is, the method of accurately emulating the user environment workload on the computer that will do the testing.

Real people
RTE (Hardware)
Keystrokes (Software)
Modeling
Subsystem Tested

Figure 3a. User Simulation Benchmarking

The most accurate method of evaluating a system for its intended use is to have the people who will utilize the system run actual programs. This is seldom practical from a cost and time standpoint. In addition, activities of real people are seldom reproducible; therefore, an accurate comparison of competing systems is not practical in a short time.

Remote Terminal Emulation (RTE) is an increasingly used method of benchmarking. The typical approach is to first capture the keystrokes and system response from a user interacting with the intended applications. The RTE computer then generates multiple versions of the captured transaction to a system under evaluation. This, however, is difficult when the objective is to emulate a multiuser, multitasking environment. RTE is the most accurate benchmarking methodology in wide use today. The cost and time for such extensive evaluation, however, can only be justified in procurement where millions of dollars are involved.

Some evaluations are conducted by capturing terminal data and loading it into the system under test. The feedback of the captured keystrokes occurs internally through software, as opposed to externally through a second system (RTE). The system under test contains both the software emulating the user activity and the software necessary to measure and report performance. The result is a system that is testing itself. However, the distortion created by this approach reduces the accuracy of this method.

Performance must be divided into two categories prior to discussing its actual measurement: speed and throughput. Speed reflects the ability of the system to perform a single task (which may be complex). Throughput is the total amount of work that a computer can do in a given amount of time.

The Modeling approach to user simulation recognizes that user activity will generate diverse system loads that utilize computer subsystems. As many as 30 to 40 individual subsystem tests may be used to measure system functions such as arithmetic, computation, disk access, and logic & memory efficiency. The individual tests can then be mixed to exercise a variety of subsystem areas, such as the disk subsystem, floating point, integer math, and memory subsystem. Subsystem mixes can be mixed again to simulate applications such as accounting, compiling, database management, scientific operations, spreadsheets, and word processing.

For workstations, it is important that subsystem tests be capable of simulating the expanding resource requirements of a multitasking environment. By increasing the consumption of resources within tasks, the workstation's resources will saturate, which eventually limits its performance.

Multiuser modeling is achieved by generating multiple instances of the application environment. System speed and throughput can only be measured by establishing the user load and mixing *prior* to running the benchmark, to accurately represent multiple users performing multiple tasks.

The Subsystem Testing approach tests the various subsystems in isolation and leaves the application and system performance decisions to the evaluator or report generator.

Actual System & Terminals
System & Test System
Actual System Only
Manual/Paper

Figure 3b. System Benchmarking Alternatives

The variables available for system simulation are listed in Figure 3b. Assembling the actual system to be evaluated, including all terminals and I/O, and testing in the alternative configurations, is the most accurate method available. However, it is also costly and time consuming.

With RTE, two systems must be used: the system under test, plus an additional system with the size and capability to run the developed workload emulation suites. This type of testing typically requires significant programming. Major porting efforts may be involved when different RTE test systems are needed.

Only the actual system under consideration is needed for the Modeling approach. The system configuration is frequently varied, and the tests rerun to compare alternative hardware and software features.

Manual or paper evaluations are a recognized method of benchmarking system configuration. One approach is to code a typical instruction sequence being considered, count the instructions, and determine the timing. However, such a sequence may test only the processing power and be no more conclusive than MIPS.

Actual
Load Mix Emulation
Load Mix Simulation
Post Test Extrapolation

Figure 3c. Programming/Application Benchmarking Alternatives

The most accurate method of measurement is achieved by running the *actual application* program, particularly if the various methods of use are exercised. A spreadsheet program involving a 100 X 100 model will place different requirements on the system under test than a 10,000 X 10,000 model. Attempting to emulate the manner in which a given program is capable of increasing its demands on the system is not an easy task.

User emulation can be accomplished by the keystroke capture technique. Sessions using actual application programs are recorded and replayed multiple times.

The *Load and Mix simulation* of an application program is a Modeling approach. That is, individual subsystem tests are used to simulate functional subsystems, which are in turn used to simulate applications. Users interacting with application programs will place loads on various parts of the system. These loads can be simulated by testing parts of the system in a manner similar to the desired requirement. Different applications can be simulated by varying the proportions of the functional subsystem tests.

Simulating the total system load is as important as simulating individual applications. Three levels of mixing are required. The system load must reflect: 1) individual application programs, 2) multiple tasks of an individual user, and 3) multiple users.

To accurately benchmark a system, the simulated system load must first be generated and then run on the system under test.

The *Extrapolation* technique sequentially tests isolated portions of the system. Although these subsystems may be tested many times, no mixing for multiple users or multiple tasks is performed prior to the testing. The performance capability of the tested system for various applications is estimated after the test, either manually or through a report generator. This is like predicting how five database queries and three word processing functions will perform simultaneously based on measuring the disk, CPU, and memory.

Figure 4 summarizes the User Simulation benchmarking alternatives predominately in use today.

System Benchmarks

To determine if a system is viable for a given set of tasks, benchmarking should be approached from the perspective of the anticipated user load. What should the evaluation approach be if the system is going to be applied across a broad variety of user loads? What if users want to perform a changing variety of applications on the system? (For example, if the user's applications include order entry, which loads the disk; SPICE simulation that loads the the CPU; and Computer Aided Manufacturing (CAM) which loads the I/O.)

User simulation benchmarks would be more difficult to use in these situations. To be effective in simulating the user, the mixing must occur *before* the test is run. Conducting the test runs after the mix creates a far more realistic load to the system under test, but it is done at the expense of flexibility. To evaluate the system performance in a wide variety of uses, it is necessary to generate a large number of mixes with multiple runs, and then corollate the results.

A more efficient method is to approach benchmarking from the perspective of measuring the system's performance capabilities. It is important, however, that such tests accurately represent the computer resource loading that will be generated by the users.

As illustrated in the previous application examples, different computing environments stress a computer system's resources in different ways. The benchmark must contain tests that stress the system's resources in the same manner as the user will experience.

A computer system is composed of subsystems. Real programs simultaneously run on different subsystem elements at any given time. For example, processing

Type	User	Program	Example
Beta	Actual	Actual	Real Systems Envioronment
Remote Terminal Emulation	Hardware Reproduction	Actual	Performance Awareness
Keystroke Capture	Software Reproduction	Actual	Lanquest & Infonetics
Modeling	Load & Mix Simulation	Simulated	Aim Workstation - Suite V Multiuser - Suite III

Figure 4. User Workload Benchmarking Methods

involves the CPU and memory, while I/O can involve memory and disk. Figure 5 shows the major subsystem areas that affect performance. Speed is a measurement of how well these subsystems work together.

In actual use, computers overlap operations to achieve efficiency. While the first program is accessing the disk, the second program can be using the processor. If multiple disks are present, multiple accesses can be made. This simultaneous activity creates a competition for resources. This activity should continue until system saturation occurs. Throughput realistically profiles potential user environments.

Processor Subsytem	This includes the CPU(s), coprocessors such as floating point unit, and cache.
Disk Subsystem	This includes multiple disks, disk controlers, and disk caching.
Memory Subsystem	Main memory, DMA devices, and memory controllers.
Terminal I/O	Intelligent terminal controllers and buffering.
Graphics Subsystem	It is difficult to provide standard tests for this increasingly important area of computer system perfomance due to the lack of standards at this time.
Compiler Optimization	The increased opportunity for improvement during compilation is too important to ignore in today's computing environment with multiple processors and RISC architectures.
Operating System	How the operating system schedules the above activities.

Figure 5. Major Computer Benchmarking Areas

Not all benchmarks have the capability to generate a simulated load and pretest mixing to represent realistic loading. Figure 6 summarizes the benchmarking concepts used in system benchmarking. Many of the popular public domain benchmarks such as Dhrystone, Whetstones, and Linpac deal with limited subsystem interaction, such as processor and memory.

One widely used benchmarking methodology is to sequentially conduct multiple tests on selected subsystems. This approach ignores the different service level requests generated by different user programs. For example, database programs treat disks differently than spreadsheets. More important, the sequential subsystem approach does not test the affect of programs simultaneously requesting service (e.g. when task #1 is processing, task #2 is using disk I/O). The conclusion of overall system speed and throughput is left to the evaluator.

Concept	Implementation	Example
Total System	Simulated Load & Mix	Aim - Suite IV
Subsystem	Multiple Instances of Single Thread	Business Benchmark
	Single Thread	Aim - Suite II
LTD Subsystem	CPU & Memory	Dhyrstones, Whetstones

Figure 6. System Benchmarking Concepts

Another standard benchmarking approach conducts single instances of multiple subsystem tests. A report generator is used to summarize various subsystem performances. Again, this approach ignores the simultaneous loading effect that can occur in today's multiuser, multitasking computer environment.

In all benchmarking tests, the programming techniques, language syntax, and subsystem loads should be similar to the programs run on real systems. Otherwise misleading tests may occur that are difficult to detect. For example, testing a system benchmark that is smaller than the cache is not representative.

The Use of Reports

There are a wide variety of reports available in the trade press and from third parties. When used for selection purposes, they allow buyers to choose a number of systems for further investigation (or alternately provide an instrument for elimination).

As with benchmarks, a variety of information is available. One report provides overall system speed and throughput, and includes subsystem data to pinpoint system bottlenecks. Another relates a number of subsystem performance results and leaves the exercise of estimating overall system results to the reader. Results published in the trade press typically address CPU and memory speed without addressing the other system's functions, such as the disk and operating system that contribute so much to speed and throughput.

- It is advisable not to depend on reports alone for system purchase. Additional evaluations should be conducted on the selected systems to determine their performance capabilities. The type of benchmark is typically determined by the dollar value of the procurement and time availability.

The Good News and Bad News

The good news is that there are a wide range of benchmarking techniques available to end users and manufacturers to assess the performance levels of today's computers. The extent to which speed and throughput are measured varies greatly. It is possible to spend significant time and effort only to end up making decisions based on limited or misleading information.

Before proceeding, determine what it is you want to measure. What accuracy do you require? Is it absolute or relative accuracy? Establish the cost and time to evaluate the number of systems you have in mind. Then use the most appropriate method.

AIM TECHNOLOGY
4699 Old Ironsides Drive, Suite 150
Santa Clara, CA 95054
Telephone: 408-748-8649
800-848-UNIX
FAX: 408-748-0161
UUCP staff@aimt.uu.net

RISC VS. CISC UNIX SYSTEM PERFORMANCE

Jim Geers, President
AIM Technology
Santa Clara, CA

Conflicting claims abound on the performance of Reduced Instruction Set Computers (RISC) compared to Complex Instruction Set Computers (CISC). Manufacturers of processor chips frequently use synthetic benchmarks such as Whetstones, Dhrystones, LINPAC and a variety of "home brew" tests to prove the mathematical prowess of their designs. Subsystem tests have been used to compare RISC and CISC systems to report that CISC beats RISC.

With all of these conflicting claims, AIM Technology set out to establish which type of system truly delivers more computing power to the UNIX user. This can be established by investigating the performance of different systems that users can purchase today. AIM's most recent tests compare the performance of five CISC and five RISC systems (see Table 1).

System Performance Rating

The first test series was used to determine how much "user horsepower" is actually delivered by each type of system. AIM's Multiuser Benchmark - Suite III - was used for this system test. This benchmark combines a series of software tests mixed to simulate both the instruction stream and the system resource loading that would be developed by multiple users on the test system. Although Suite III is capable of simulating a variety of application environments, a general test mix was used for this series of tests as it represents a middle ground betwen scientific and business applications.

TABLE 1. SYSTEMS TESTED

Type	System	CPU	Clock	Flt Point	RAM	Disk	OS	Date Tested
RISC								
	HP 9000/835	HP-PA	15Mhz	Integrated	24Mb	550Mb(x4)	HP-UX 3.0	11/88
	Intergraph	Clipper	40Mhz	Integrated	16Mb	350Mb	System V	11/88
	Sun 4/110	SPARC	14.2Mhz	Weitek1164/5	8Mb	356Mb	Sun OS 4.0	11/88
	MIPS M/120	R2000	16.7Mhz	R2010	16Mb	330Mb(x2)	RISC/os 3.1	11/88
	MIPS M2000	R3000	25Mhz	R3010	64Mb	850Mb(x2)	UMIPS 3.1	10/88
CISC								
	Convergent PC200	80386	20Mhz	Weitek 1167	8Mb	145Mb	CTIX/386(SVR)	4/88
	Intel SYP302	80386	25Mhz	80387	8Mb	380Mb	Sys V/386 3.1V2	6/88
	Sun Roadrunner (386i)	80386	25Mhz	80387	16Mb	327Mb	Sun OS 4.0	6/88
	Moto 3600 Dept Com Sys	68030	25Mhz	MC68882	12Mb	390Mb(x4)	Sys V68 R3	6/88
	Moto 3600 Workgroup Sys	68030	25Mhz	MC68882	8Mb	300Mb(x2)	Sys V68 R3	6/88

TABLE 2. AIM GENERAL TEST MIX
MULTIUSER TESTING

Functional Area	Subsystem Tests
20% RAM	RAM write short, long RAM read short, long, character
10% Float	float add, multiply, divide double add, multiply, divide
20% Disk	disk write, read, copy; create close calls; directory search
20% Math	short, add, multiply, divide long add, multiply, divide
20% Logic	call func ();call func (int); call func (int, int); call func (3*int)
10% Pipe	pipe copy

AIM's General Test Mix is sumarized in Table 2. Suite II generates multiple tests, each representing a potential system user. Performance of the system under test is measured for increasing numbers of users until system response time becomes excessive.

As a result of this Multiuser Benchmark testing, each system is given a performance rating based on its relative performance. The performance rating is a percentage relative to the performance of a control, or "normalized," system, referred to as the "Standard AIM System." For ease of comparison the Standard AIM System was selected such that most VAX 11/780's test to 100%.

Table 3 shows the average of each test group's AIM Performance Rating. Test results summarized throughout this analysis that involve normalized numbers have been averaged using the harmonic mean. This shows that the average RISC system tested almost 7 times the level of a VAX while the average CISC system tested only 3 times the level of a VAX.

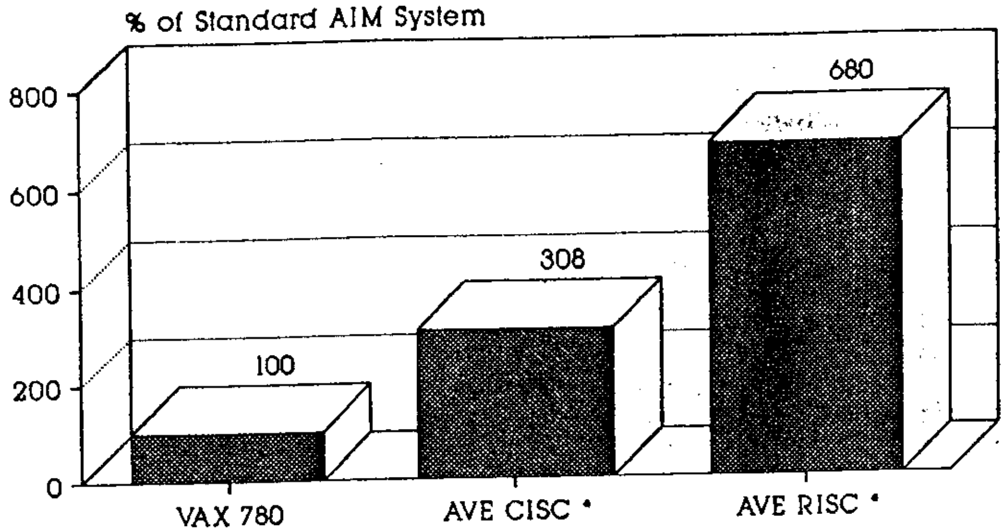
System Workload Rating

The User Load Rating is an indication of the tested system's multitasking throughput capability. This rating can be equated to either a group of general users actively using the system or by a few users engaged in very heavy computation. The control, or "normalized" standard was set at the response rate achieved, on a typical VAX 11/780 with 12 users. Each test groups User Load Rating is shown in Table 4. This shows that the average RISC system tested could handle a user load over double that for the average CISC systems.

Understanding System Performance

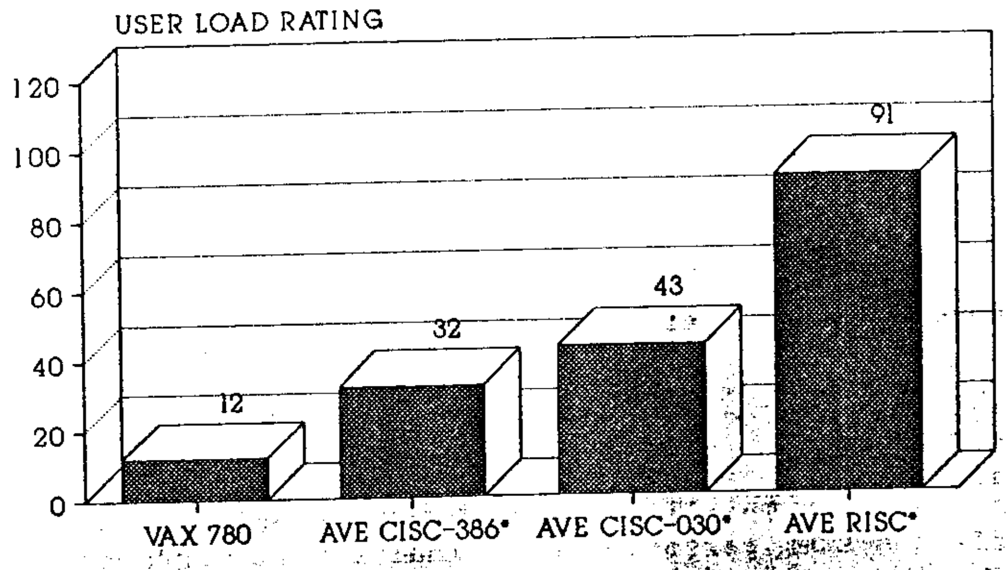
System performance is based on more than processor architecture and clock rate. AIM has found in previous testing that the relative performance of five UNIX systems, all using 16.7 MHZ 68020's, varied from 70% of a VAX 780, to 218% This indicates there are other parameters that effect system performance. Final system performance will usually depend on the surrounding subsystem hardware and operating system software as well as the processor itself.

TABLE 3. PERFORMANCE RATING
RISC VS. CISC UNIX SYSTEMS



* Harmonic Mean

TABLE 4. SYSTEM THROUGHPUT
RISC VS CISC UNIX SYSTEMS



* Arithmetic Mean

The additional factors that significantly effect performance include the memory implementation, disk subsystem performance, co-processor availability, if any, and efficiency of the UNIX operating system kernel. For example, in another test, five different versions of UNIX where tested on the same hardware system. In this case, AIM found that just running different versions of UNIX had a significant effect on performance. Results at the high end indicated the throughput capability of 25 user loads, while at the low end, only 7 user loads- same hardware, just different versions of the operating system.

DIFFERENCES DUE TO IMPLEMENTATION

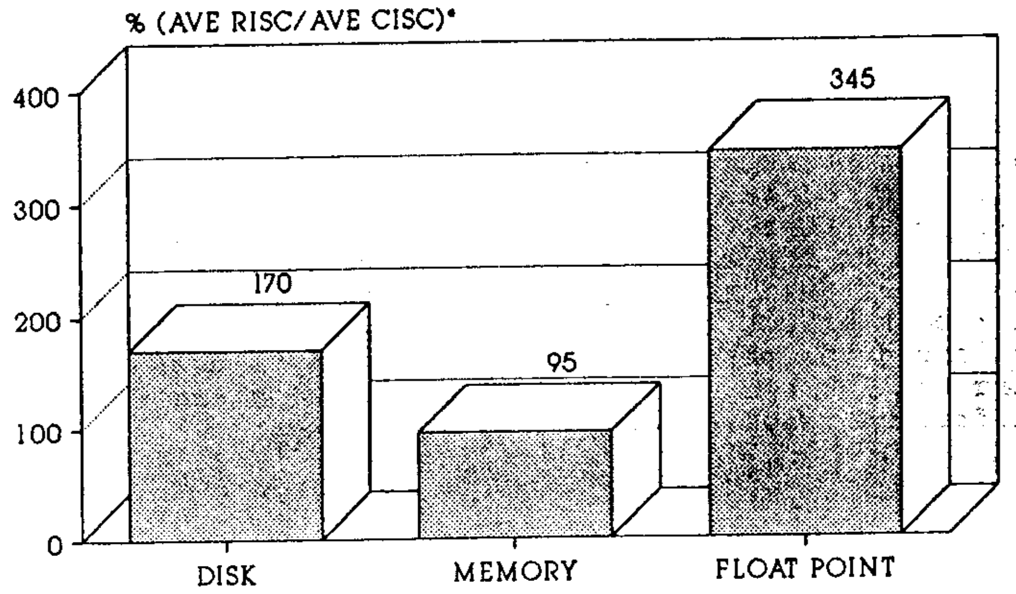
The subsystems surrounding the processor of the tested systems varied greatly. Actual configurations are shown in Table 1. Because subsystem range can vary individual system results, a second series of tests were conducted using AIM's Subsystem Benchmark - Suite II.

Suite II contains thirty-seven tests of the most frequently used performance-predicting system functions, such as disk and memory. These tests individually measure system functions associated with a wide range of subsystem elements. Taken individually, they are interesting pieces of information, much like hardware specifications. What becomes significant is the system insight gained by combining a number of these tests into groups that represent functional areas of system operation. This can give us an indication as to the contribution of that subsegment to overall system performance.

Suite II's functional tests are divided into two general categories. Table 5 compares the subsystem performance of each test group in three significant areas; disk, memory and floating point. These areas generally depend very little on the processor for performance, but contribute to overall system performance.

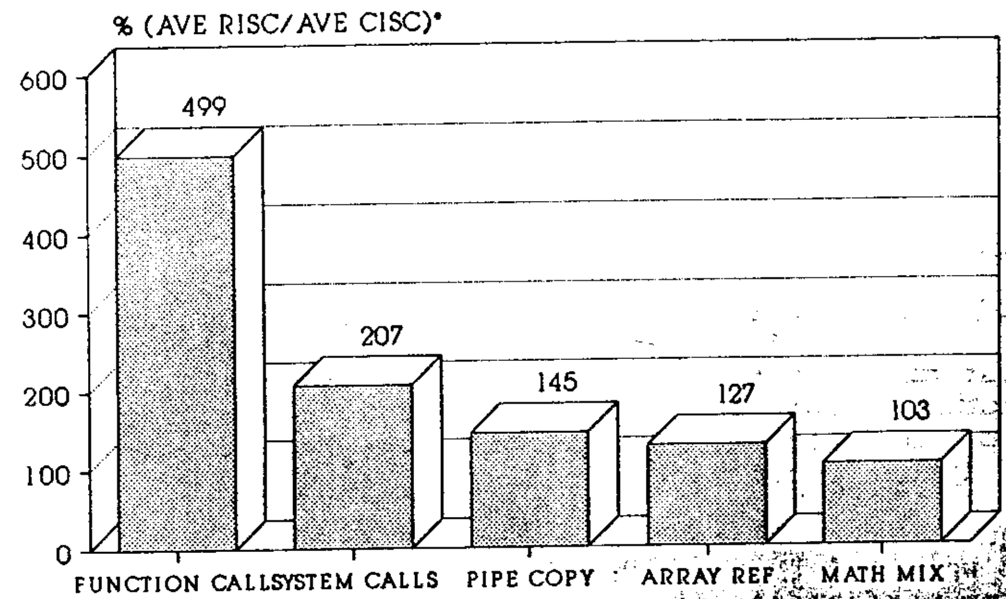
Although the overall tested system performance of the RISC systems was 221% that of the CISC systems, both the disk subsystem performance at 170% and the memory subsystem at 95% were lower than the overall system level. The floating point performance for the RISC systems was 3.45 times faster, but floating point was only 10% of the test mix (see Table 1.)

TABLE 5. SUBSYSTEM PERFORMANCE



* Harmonic Mean

TABLE 6. OTHER FUNCTIONAL AREAS



* Harmonic Mean

The second group of Suite II functional tests depend on the processor for performance, but have other significant contribution considerations. The five test areas shown in Table 6, include function calls, system calls, pipe copy, array reference timers and math mix. All tend to use both the processor and memory, and in addition, are dependent upon the operating system kernel for their performance.

CONCLUSION

The RISC systems tested appear to offer users better performance than their CISC counterparts. It appears in the near term that purchasers of multiuser systems are faced with a trade off between performance and program compatibility with a wide range of third party software vendors. If they are engaged in engineering or scientific applications and using primarily proprietary software, the increased performance of RISC systems will be very attractive. If they are engaged in business applications and depend on third party software, the CISC will be attractive.

The conclusion may be simpler for most single user systems. At three times the power of a Vax/780, CISC systems seem to deliver more than enough capability for the bulk of today's requirements.

As to performance of the processors, in view of relatively equal memory performance between the two types of systems, the results indicate that the RISC processors deliver more capability than their CISC counterparts. Higher speed disk and floating point co-processors helped the overall system performance of the RISC systems, but not enough to account for the difference. Better operating system kernels and optimizing compilers could also assist the tested RISC system.

Other considerations such as price, technical support and maintenance services need to be taken into account in a procurement evaluation.

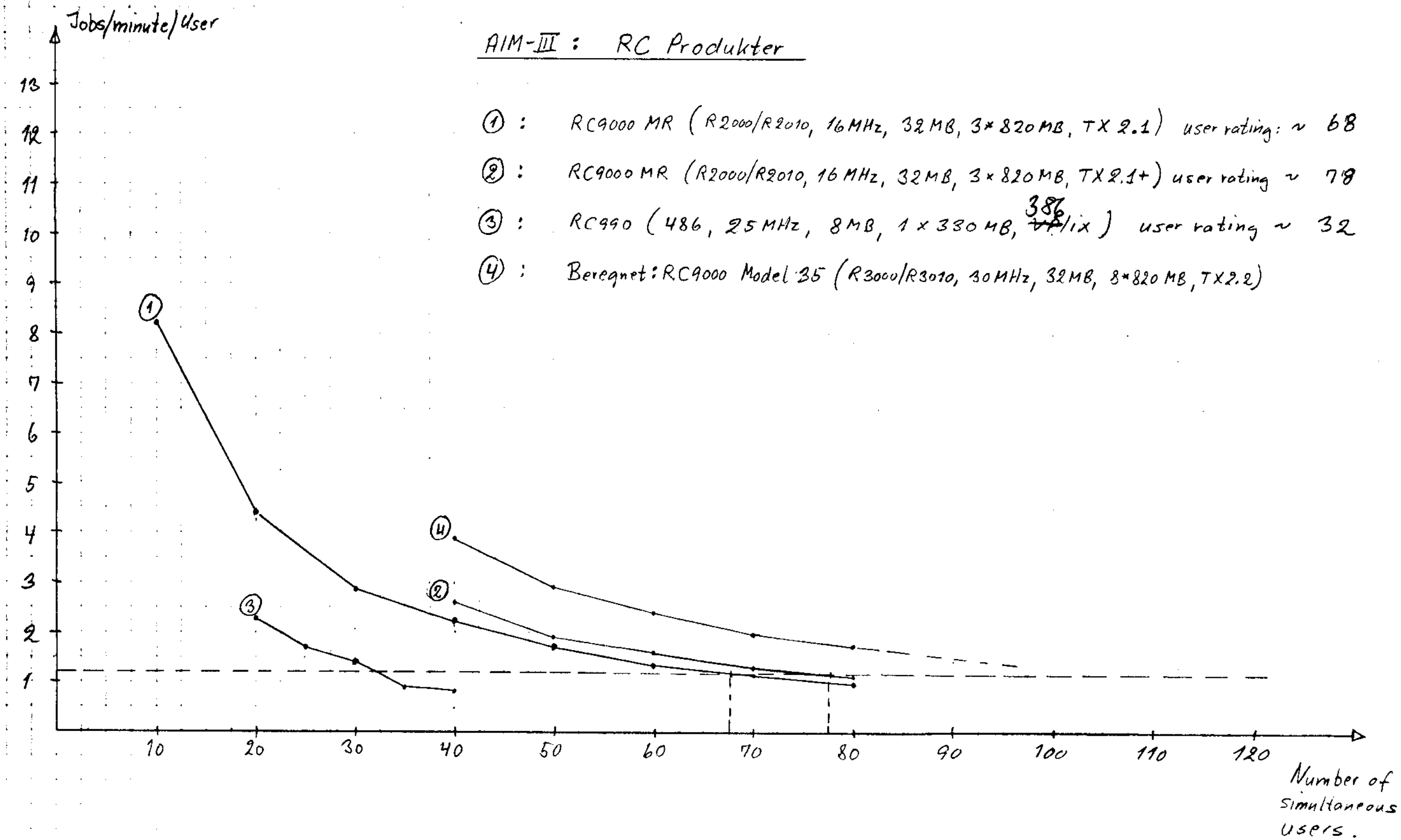
INDIVIDUAL TEST RESULTS

All tests were conducted by AIM Technology personnel on systems provided by the manufacturers.
Each manufacturer had the opportunity to review the test results. Individual results are not disclosed without the manufacturer's authorization. System test results that have been released at this time are listed in Table 7. AIM Performance Reports are available on individual systems.

TABLE 7. INDIVIDUAL SYSTEM TEST RESULTS
(VAX 780 = 100% and 12 user load)

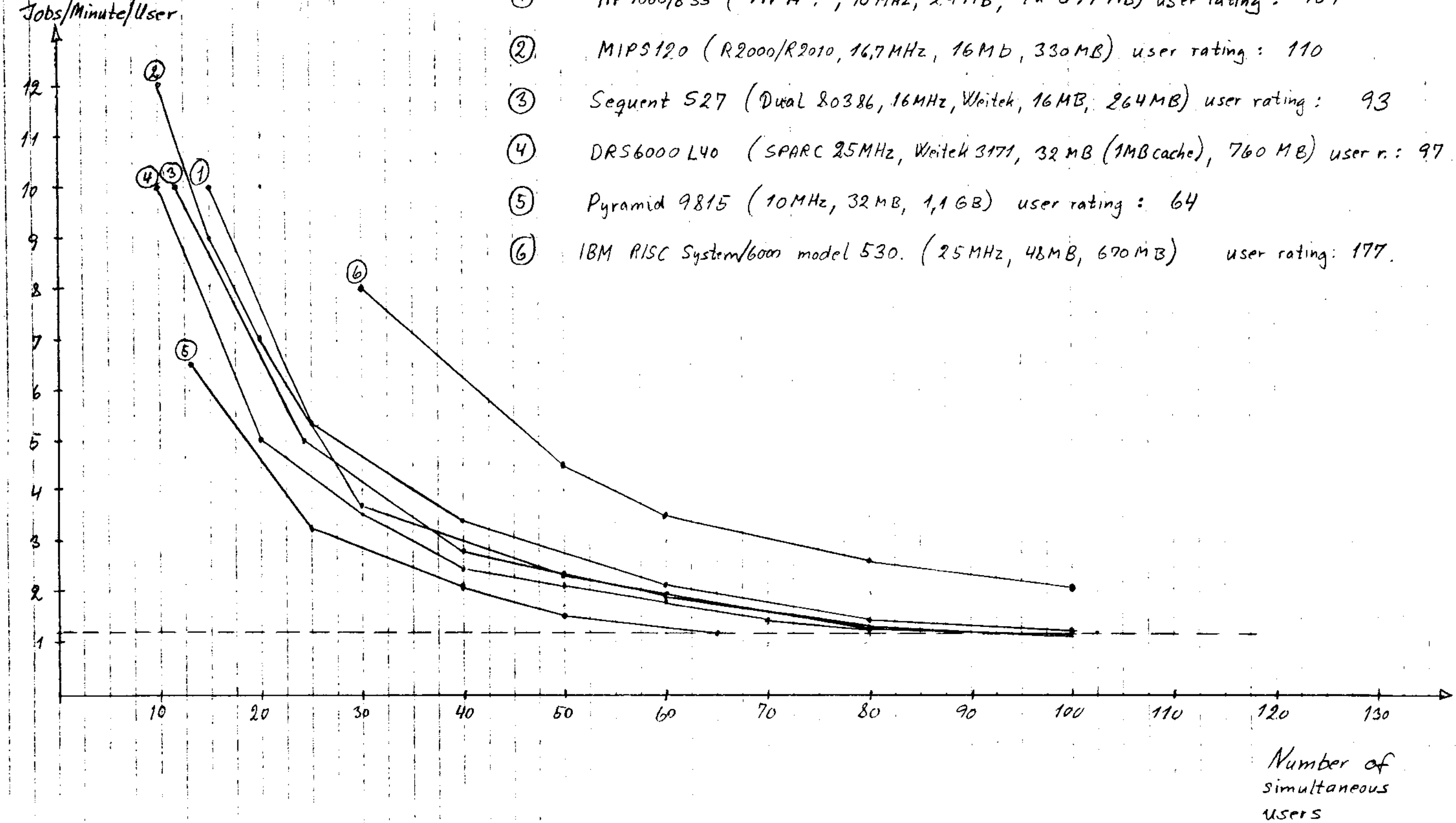
TYPE/SYSTEM	PERFORMANCE	USER LOAD
RISC		
HP 9000/835	1004%	109
INTEGRAPH	N/A	N/A
SUN 4/110	457%	52
MIPS M120	989%	110
MIPS M2000	N/A	N/A
CISC-80386		
CONVERGENT PC200	376%	38
INTEL SYP302	246%	25
SUN ROADRUNNER 3861	313%	35
CISC-68030		
MOTO 3600 DEPT	398%	46
MOTO 3600 WORKGROUP	400%	40

N/A -not released by manufacturer


AIM-III : RC Produkter
1 : RC9000 MR (R2000/R2010, 16 MHz, 32 MB, 3*820 MB, TX 2.1) user rating: ~ 68
2 : RC9000 MR (R2000/R2010, 16 MHz, 32 MB, 3 × 820 MB, TX 2.1+) user rating ~ 78
3 : RC990 (486, 25 MHz, 8 MB, 1 × 330 MB, 386/ix) user rating ~ 32
4 : Beregnet: RC9000 Model 35 (R3000/R3010, 30 MHz, 32 MB, 8*820 MB, TX 2.2)
Jobs/minute/User
13
12
11
10
9
8
7
6
5
4
3
2
1
10
20
30
40
50
60
70
80
90
100
110
120
Number of simultaneous users.

AIM-III : Konkurrent produkter.

1. HP9000/835 (HPA . , 15MHz, 24MB, 4* 571 MB) user rating : 109
2. MIPS120 (R2000/R2010, 16,7MHz, 16Mb, 330MB) user rating : 110
3. Sequent S27 (Dual 80386, 16MHz, Weitek, 16MB, 264MB) user rating : 93
4. DRS6000 L40 (SPARC 25MHz, Weitek 3171, 32 MB (1MB cache), 760 MB) user r. : 97
5. Pyramid 9815 (10MHz, 32MB, 1,1 GB) user rating : 64
6. IBM RISC System/6000 model 530. (25MHz, 48MB, 670MB) user rating: 177.



RC9000 MR

Office mix.

Spreadsheet ↓ wordprocessing ↙

```
sa9000 wp Apr 19 16:52 TTY OFF, TAPE OFF, LP OFF, VM OFF, 30 sp 70 wp
ua9000 wp Apr 19 16:53 TTY OFF, TAPE OFF, LP OFF, VM OFF, 30 sp 70 wp
 1573.6 control 0.031775 proc/sec 2008.9 real 910.4 cpu    40
 1701.6 control 0.029384 proc/sec 2243.3 real 1036.0 cpu   45
 1775.7 control 0.028159 proc/sec 2465.9 real 1152.5 cpu   50
 2117.1 control 0.023617 proc/sec 2766.0 real 1280.5 cpu   55
 2253.9 control 0.022183 proc/sec 2995.5 real 1391.7 cpu   60
 2384.8 control 0.020966 proc/sec 3245.7 real 1513.4 cpu   65
 2479.6 control 0.020165 proc/sec 3498.7 real 1636.8 cpu   70
 2682.9 control 0.018636 proc/sec 3696.6 real 1742.2 cpu   75
 2835.8 control 0.017632 proc/sec 3878.6 real 1850.8 cpu   80
```

antal brugere.

A I M III

A N D R E L E V E R A N D Ø R E R S P R O D U K T E R .

AIM Performance Report™

A UNIX™ Performance Summary Based on the AIM Benchmarks.

Sequent S27
(2-processor system)

Performance Rating: 738%
User Load Rating: 93

System Configuration Tested:

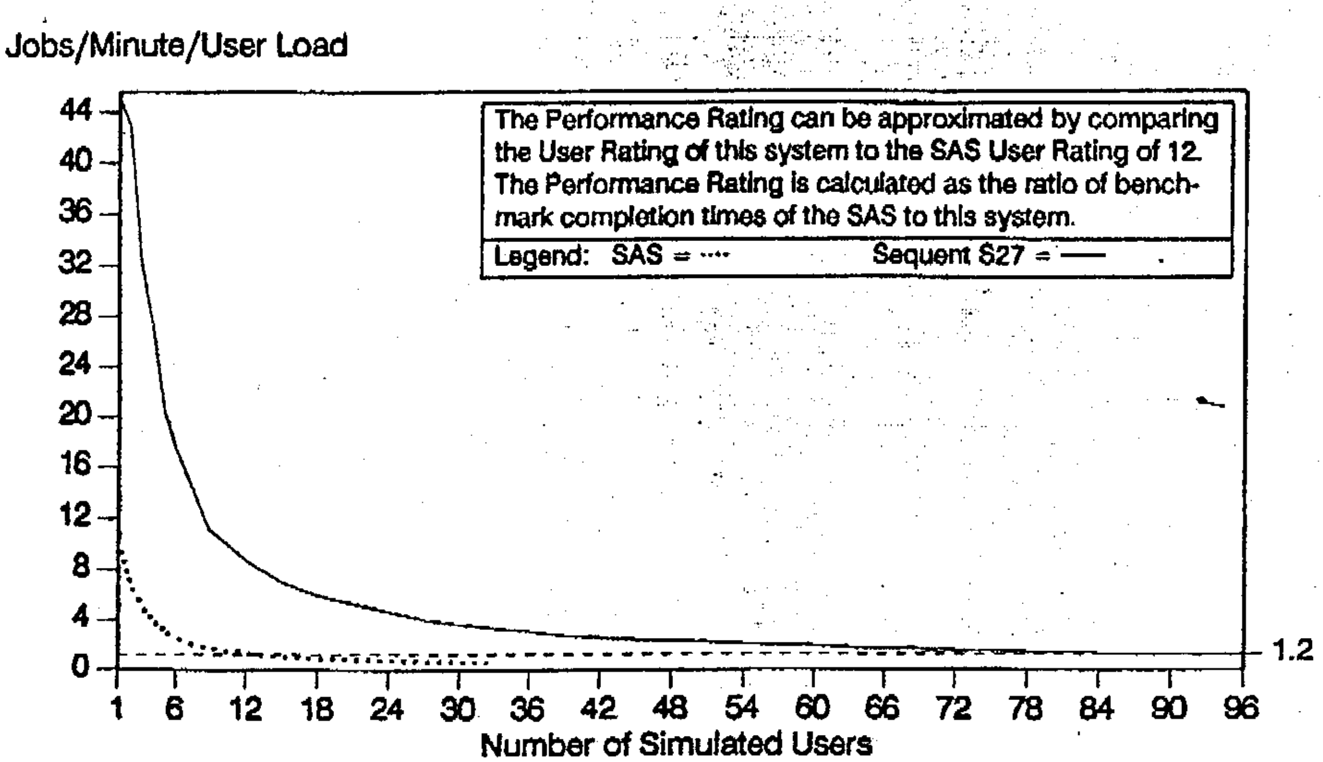
CPU	80386(2)
Clock	16MHz
Floating Point	Weitek(2)
RAM	16 Mb
Disk	264 Mb
O/S	DYNIX 3.0.12
Date Tested	28 March 1989

The **Performance Rating** reflects the overall performance of this system, normalized to the Standard AIM System (SAS).† The Performance Ratings of a wide range of UNIX systems can be compared using available AIM Performance Reports.

The **User Load Rating** indicates the multitasking user load where the system's performance can become unacceptable.

Work Throughput

Work throughput as a function of the simulated user load is shown below. AIM uses 1.2 jobs per minutes per user as a reference point. The actual number of users a system may accommodate will vary with the type of use and physical connections.



† Most VAX™ 11/780 configurations will typically rate 100% (and 12 users) of the Standard AIM System.

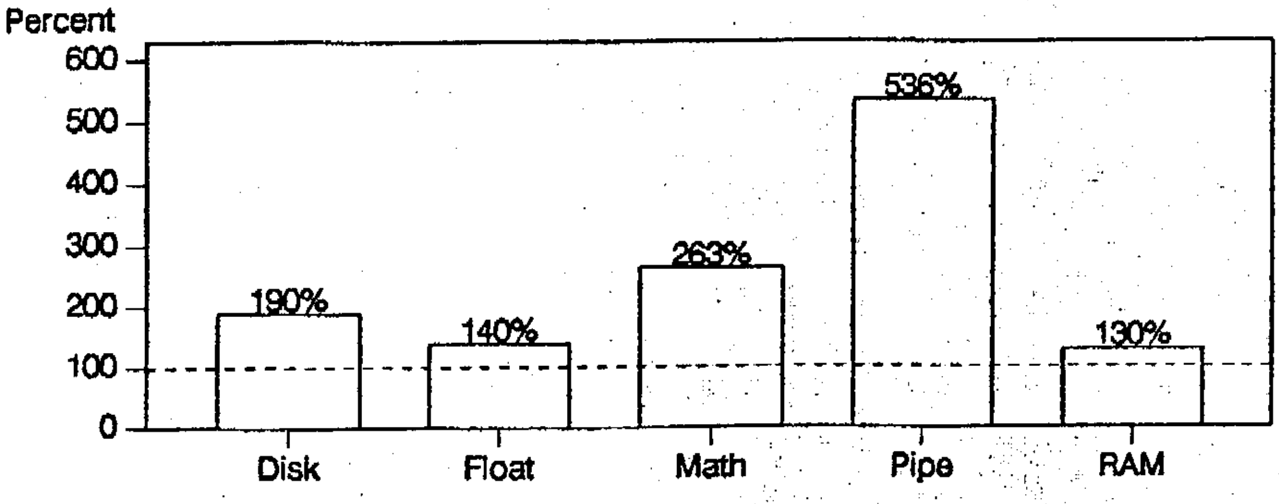
AIM Technology
4699 Old Ironsides Dr., Ste. 150
Santa Clara, CA 95054
408-748-8649

AIM Performance Report

Sequent S27 / 2

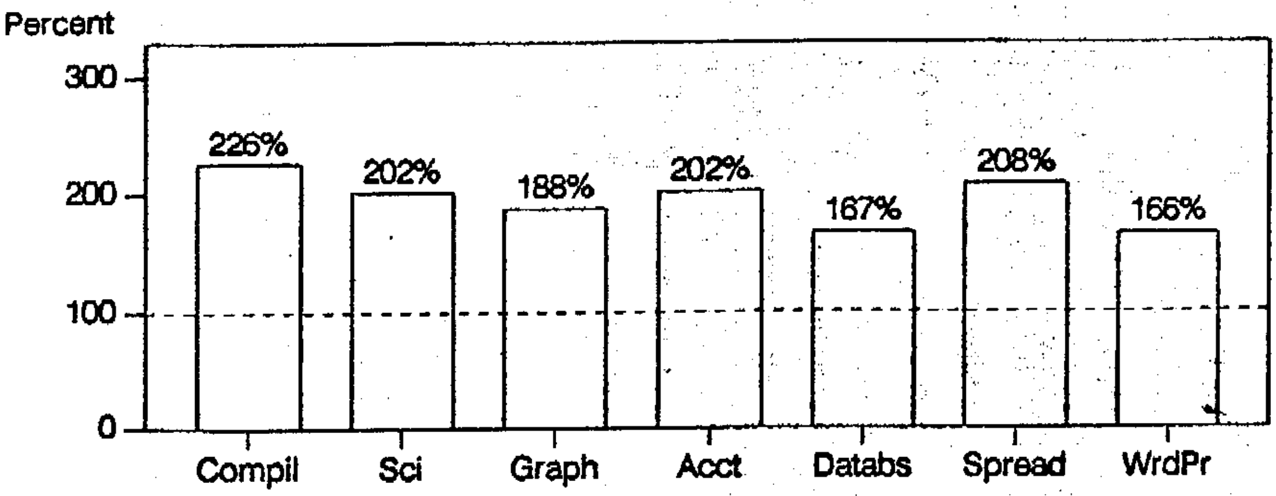
Subsystem Performance

Single-processor performance in five subsystem areas is shown below. Scores are normalized to the Standard AIM System. See the APR Supplement for more information on the subsystem tests.



Application Performance

Performance variations in typical engineering and business applications are shown below. Scores are normalized to the Standard AIM System. See the APR Supplement for more information on the application tests.



This report provides a brief performance summary using the AIM application and multiuser benchmarks. System performance will vary according to configuration, application mix and usage. More detailed analysis of these variables can be obtained with the AIM Benchmark™ Suites.

AIM Technology is the industry leader in providing benchmarking tools for UNIX systems, and also provides AIM Job Scheduler™, AIM Disk Tuner™, and AIM Job Accounting™.

For information on other AIM Performance Reports and the AIM Benchmarks, call AIM at 408-748-8649.

AIM Performance Report™

A UNIX™ Performance Summary Based on the AIM Benchmarks.

IBM RISC System/6000 Model 530

Performance Rating: 1627%
User Load Rating: 177

System Configuration Tested:

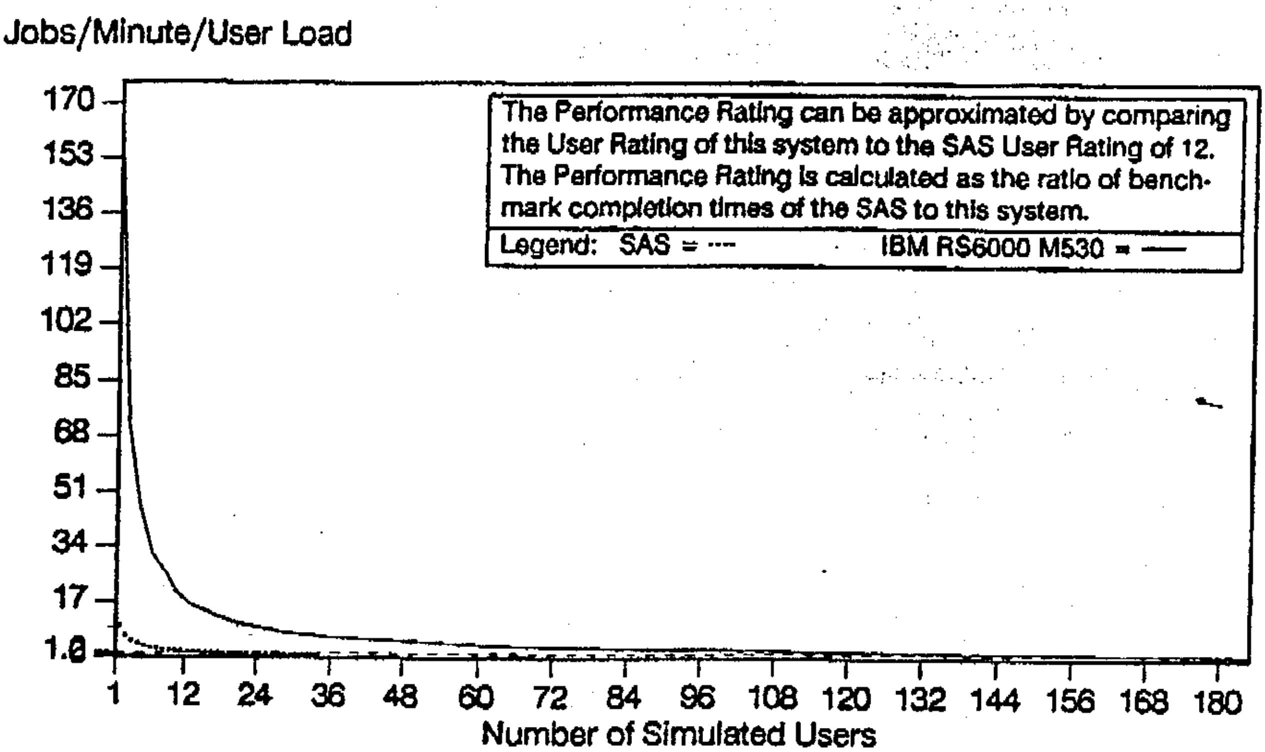
CPU	RISC Power
Clock	25 MHz
Floating Point	In Power Set
RAM	48 MB
Disk	670 MB
O/S	AIX v3.1 (eng. level)
Date Tested	March 29, 1990

The **Performance Rating** reflects the overall performance of this system, normalized to the Standard AIM System (SAS).† The Performance Ratings of a wide range of UNIX systems can be compared using available AIM Performance Reports.

The **User Load Rating** indicates the multitasking user load where the system's performance can become unacceptable.

Work Throughput

Work throughput as a function of the simulated user load is shown below. AIM uses 1.2 jobs per minutes per user as a reference point. The actual number of users a system may accommodate will vary with the type of use and physical connections.



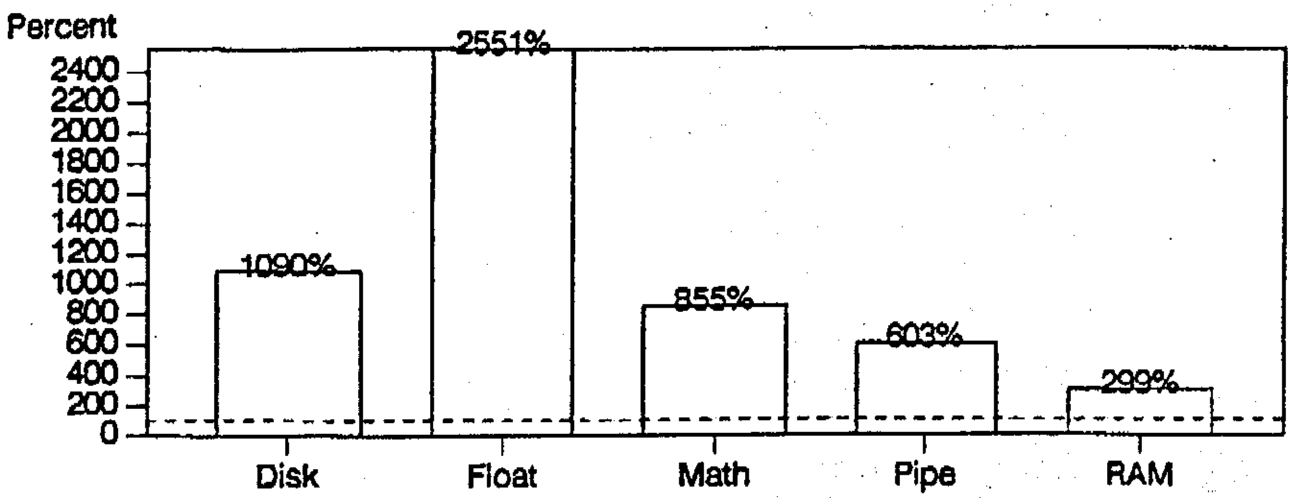
† Most VAX™ 11/780 configurations will typically rate 100% (and 12 users) of the Standard AIM System.

AIM Technology
4699 Old Ironsides Dr., Ste. 150
Santa Clara, CA 95054

AIM Performance Report IBM RS6000 M530

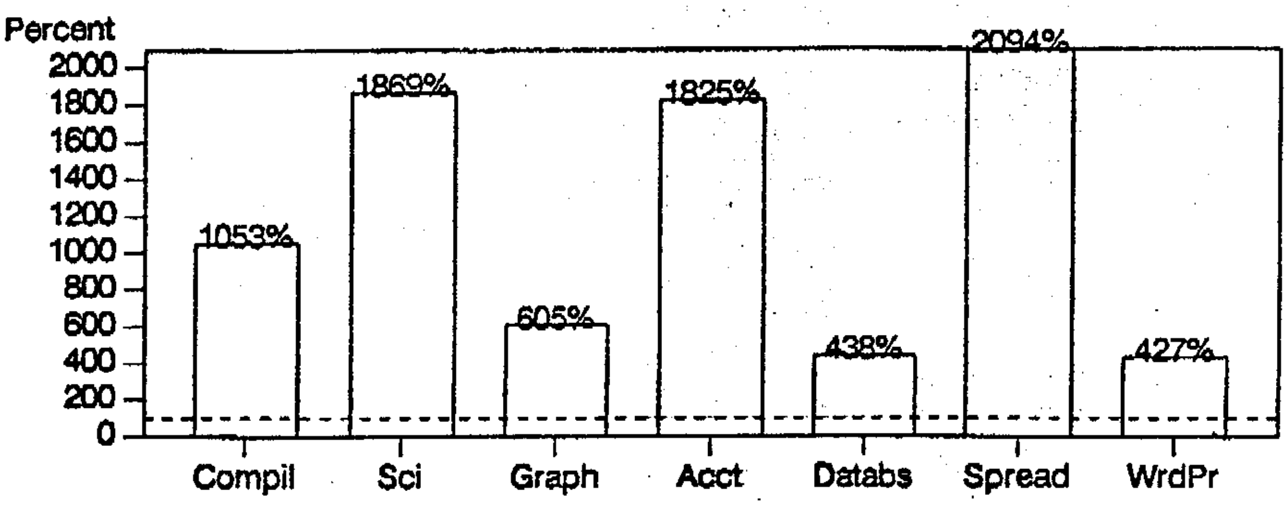
Subsystem Performance

Single-processor performance in five subsystem areas is shown below. Scores are normalized to the Standard AIM System. See the APR Supplement for more information on the subsystem tests.



Application Performance

Performance variations in typical engineering and business applications are shown below. Scores are normalized to the Standard AIM System. See the APR Supplement for more information on the application tests.



This report provides a brief performance summary using the AIM application and multiuser benchmarks. System performance will vary according to configuration, application mix and usage. More detailed analysis of these variables can be obtained with the AIM Benchmark™ Suites.

AIM Technology is the industry leader in providing benchmarking tools for UNIX systems, and also provides AIM Job Scheduler™, AIM Disk Tuner™, and AIM Job Accounting™.

For information on other AIM Performance Reports and the AIM Benchmarks, call AIM at 408-748-8649.

AIM Performance Report™

A UNIX™ Performance Summary Based on the AIM Benchmarks.

DRS 6000L40

Performance Rating:	**878%**
User Load Rating:	**97**

System Configuration Tested:

CPU	CY7C601 SPARC
Clock	25 MHz
Floating Point	Weitek 3171
RAM	32 MB(1MB cache)
Disk	760 MB(2)
O/S	DRS/NX V4.L0l61(SysVR4)
Date Tested	January 6, 1990

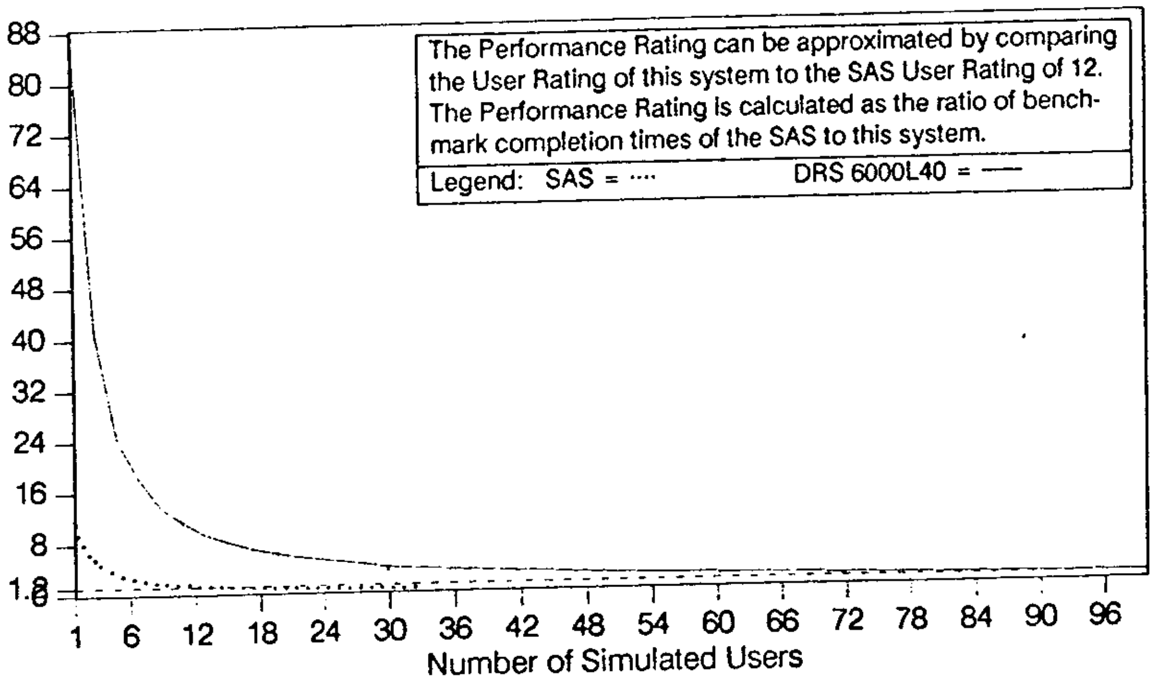
The **Performance Rating** reflects the overall performance of this system, normalized to the Standard AIM System (SAS).† The Performance Ratings of a wide range of UNIX systems can be compared using available AIM Performance Reports.

The **User Load Rating** indicates the multitasking user load where the system's performance can become unacceptable.

Work Throughput

Work throughput as a function of the simulated user load is shown below. AIM uses 1.2 jobs per minutes per user as a reference point. The actual number of users a system may accommodate will vary with the type of use and physical connections.

Jobs/Minute/User Load



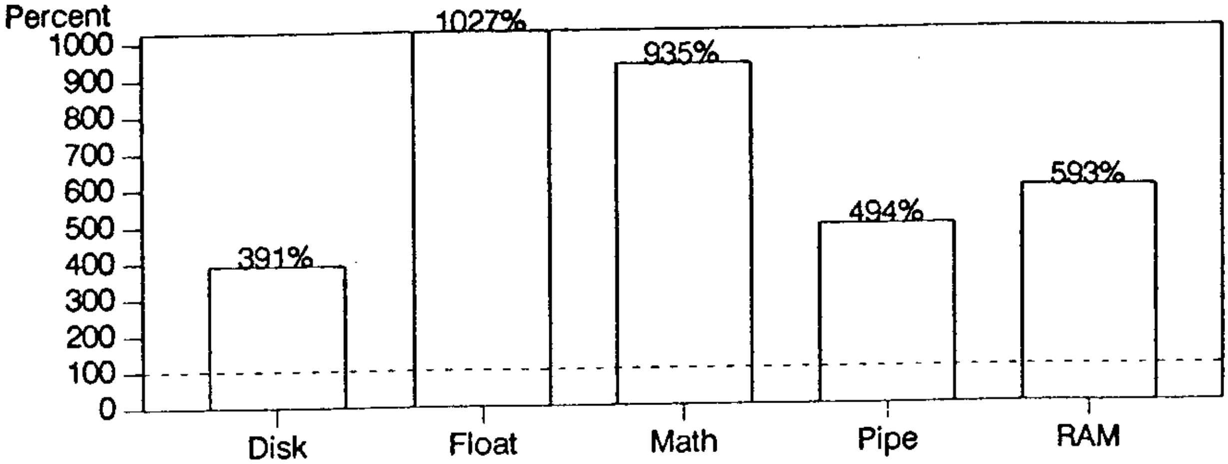
† Most VAX™ 11/780 configurations will typically rate 100% (and 12 users) of the Standard AIM System.

AIM Technology
4699 Old Ironsides Dr., Ste. 150
Santa Clara, CA 95054

AIM Performance Report DRS 6000L40

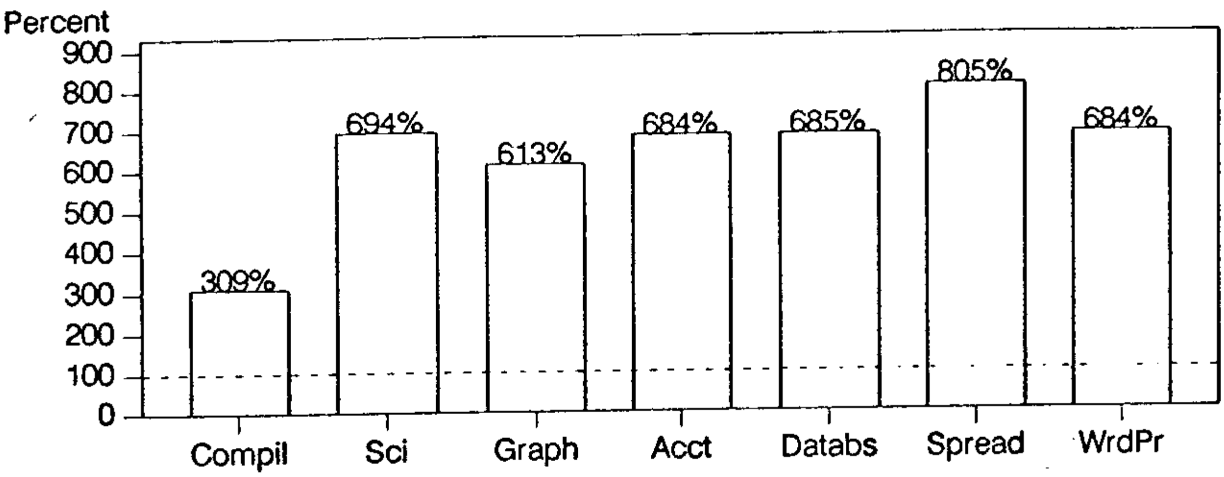
Subsystem Performance

Single-processor performance in five subsystem areas is shown below. Scores are normalized to the Standard AIM System. See the APR Supplement for more information on the subsystem tests.



Application Performance

Performance variations in typical engineering and business applications are shown below. Scores are normalized to the Standard AIM System. See the APR Supplement for more information on the application tests.



This report provides a brief performance summary using the AIM application and multiuser benchmarks. System performance will vary according to configuration, application mix and usage. More detailed analysis of these variables can be obtained with the AIM Benchmark™ Suites.

AIM Technology is the industry leader in providing benchmarking tools for UNIX systems, and also provides AIM Job Scheduler™, AIM Disk Tuner™, and AIM Job Accounting™.

For information on other AIM Performance Reports and the AIM Benchmarks, call AIM at 408-748-8649.

AIM Performance Report™

A UNIX™ Performance Summary Based on the AIM Benchmarks.

MIPS M120	
Performance Rating:	989%
User Rating:	110

System Configuration Tested:

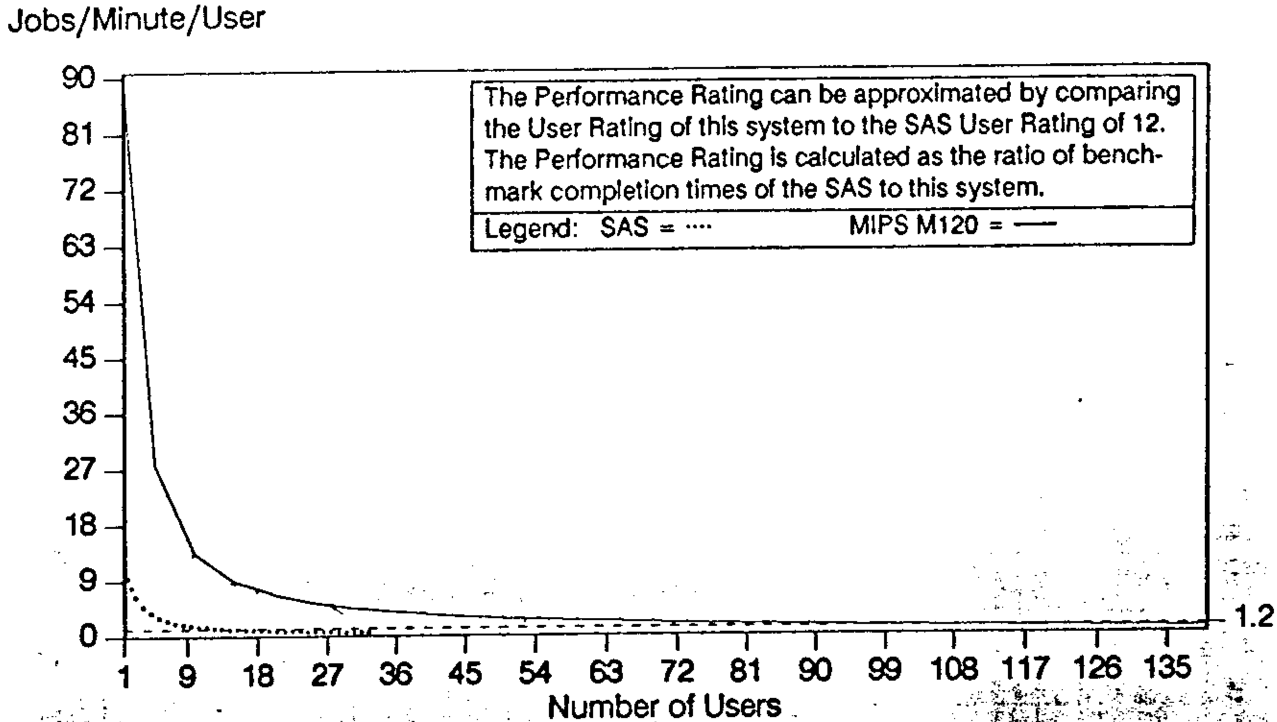
CPU	R2000
Clock	16.7MHz
Floating Point	R2010
RAM	16Mb
Disk	330Mb (2)
O/S	UMIPS 3.1
Date Tested	Oct 31, 1988

The **Performance Rating** reflects the overall performance of this system, normalized to the Standard AIM System™ (SAS).† The Performance Ratings of a wide range of UNIX systems can be compared using available AIM Performance Reports.

The **User Rating** indicates the number of active users where the system's performance can become unacceptable.

Work Throughput

Work throughput as a function of the number of active users is shown below. AIM uses 1.2 jobs per minutes per user as a reference point to determine the User Rating. The actual number of users a system may accommodate will vary with the load and type of use.



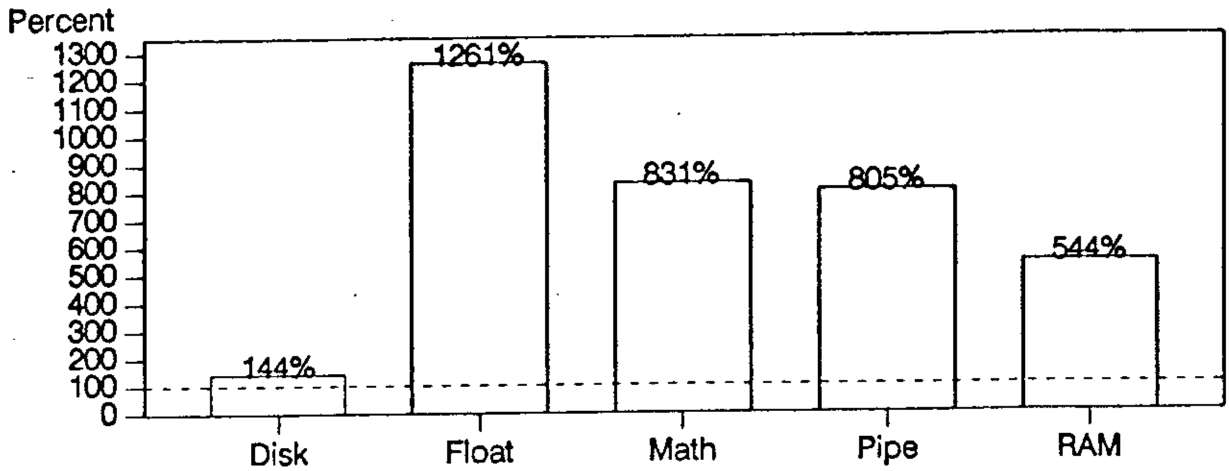
† Most VAX™ 11/780 configurations will typically rate 100% (and 12 users) of the Standard AIM System.

AIM Technology
4699 Old Ironsides Dr., Ste. 150
Santa Clara, CA 95054
408-748-8649

AIM Performance Report — MIPS M120

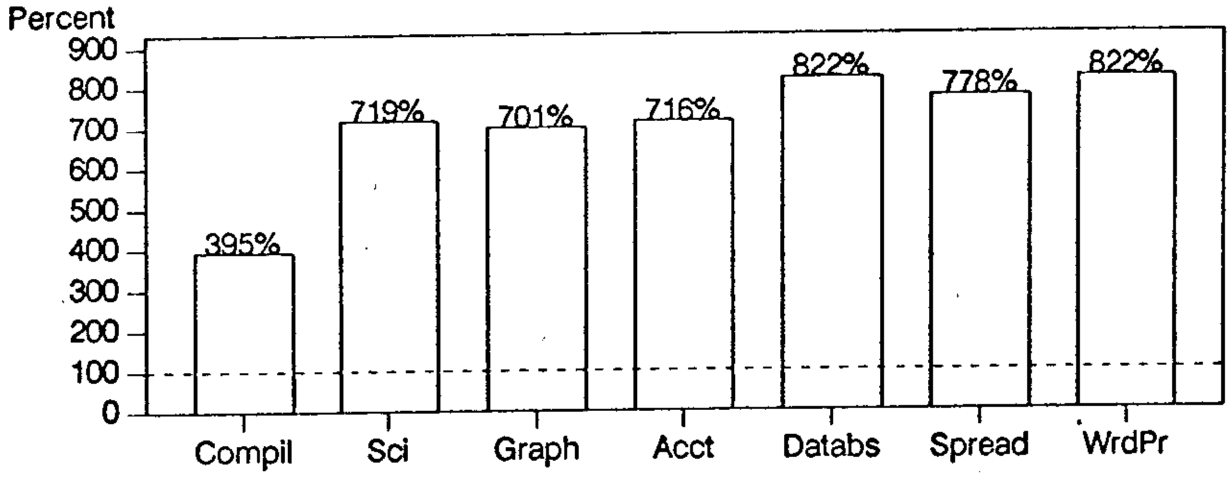
Subsystem Performance

Performance in 5 subsystem areas is shown below. Scores are normalized to the AIM Standard System. See the APR Supplement for more information on the subsystem tests.



Application Performance

Performance variations in typical engineering and business applications are shown below. Scores are normalized to the AIM Standard System. See the APR Supplement for more information on the application tests.



This report provides a brief performance summary using the AIM application and multiuser benchmarks. System performance will vary according to configuration, application mix and usage. More detailed analysis of these variables can be obtained with the AIM Benchmark™ Suites.

AIM Technology is the industry leader in providing benchmarking tools for UNIX systems, and also provides AIM Job Scheduler™, AIM Disk Tuner™, and AIM Job Accounting™.

For information on other AIM Performance Reports and the AIM Benchmarks, call AIM at 408-748-8649.

AIM Performance Report™

A UNIX™ Performance Summary Based on the AIM Benchmarks.

Hewlett-Packard 9000/835	
Performance Rating:	1004%
User Rating:	109

System Configuration Tested:

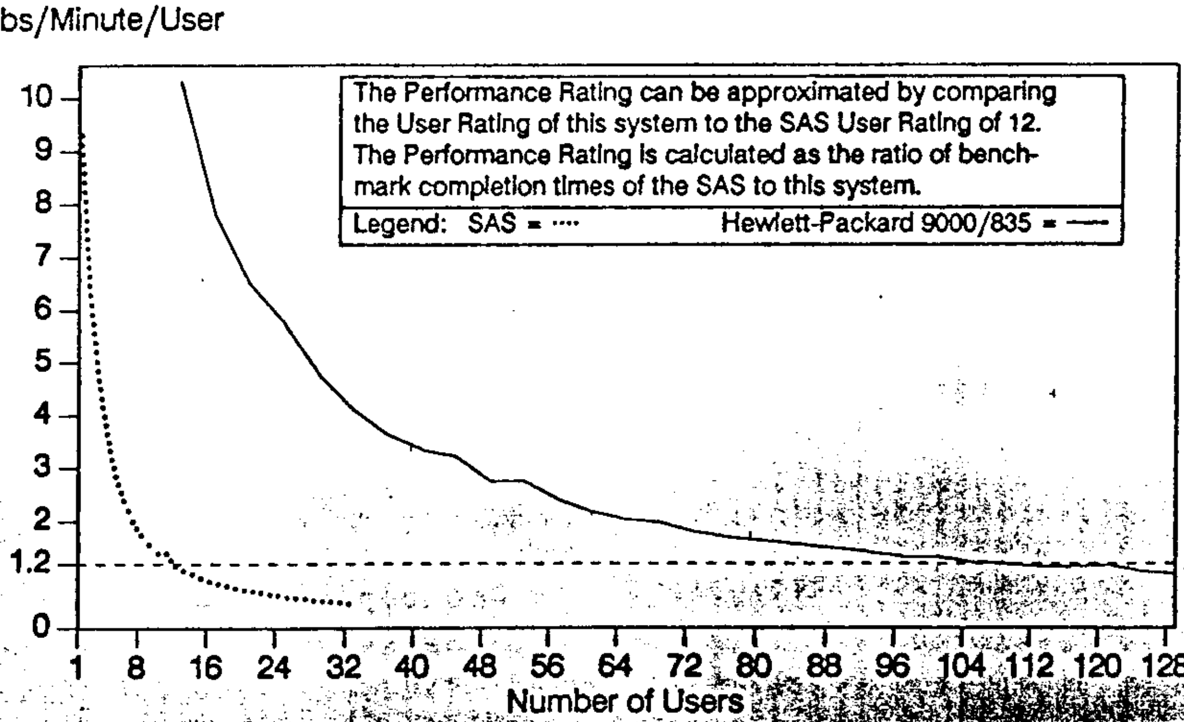
CPU	Proprietary
Clock	15 MHz
Floating Point	Proprietary
RAM	24 MB
Disk	571 MB (4)
O/S	HP/UX
Date Tested	14 Nov 88

The **Performance Rating** reflects the overall performance of this system, normalized to the Standard AIM System™ (SAS).† The Performance Ratings of a wide range of UNIX systems can be compared using available AIM Performance Reports.

The **User Rating** indicates the number of active users where the system's performance can become unacceptable.

Work Throughput

Work throughput as a function of the number of active users is shown below. AIM uses 1.2 jobs per minutes per user as a reference point to determine the User Rating. The actual number of users a system may accommodate will vary with the load and type of use.



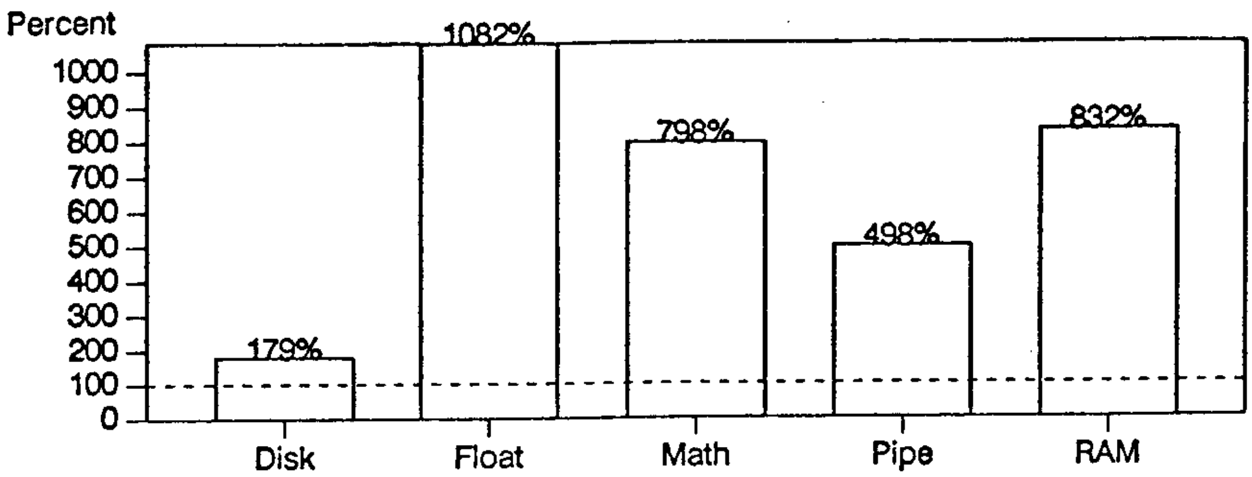
† Most VAX™ 11/780 configurations will typically rate 100% (and 12 users) of the Standard AIM System.

AIM Technology
4699 Old Ironsides Dr., Ste. 150
Santa Clara, CA 95054
408-748-8649

AIM Performance Report Hewlett-Packard 9000/835

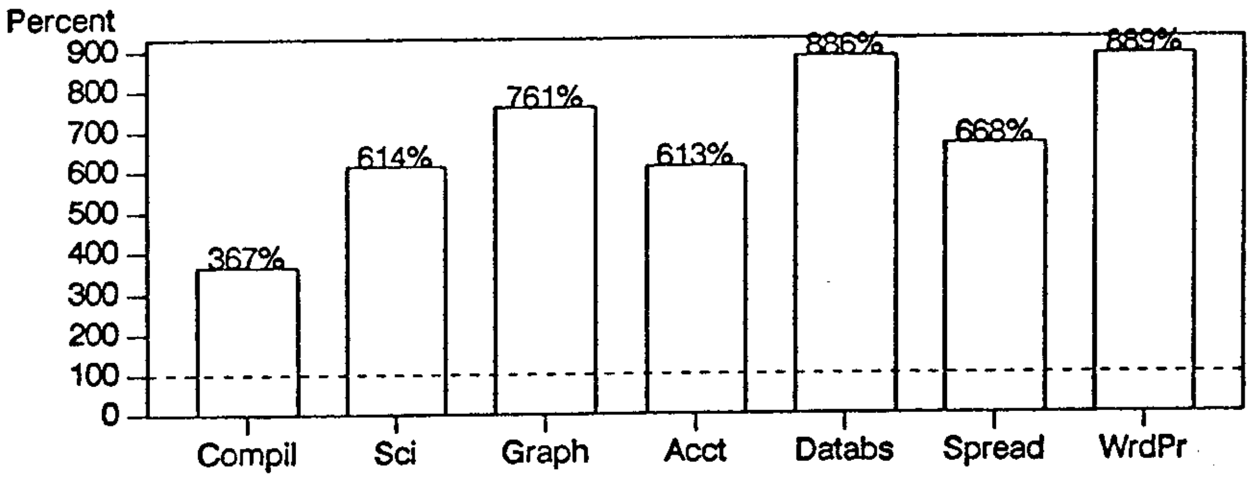
Subsystem Performance

Performance in 5 subsystem areas is shown below. Scores are normalized to the AIM Standard System. See the APR Supplement for more information on the subsystem tests.



Application Performance

Performance variations in typical engineering and business applications are shown below. Scores are normalized to the AIM Standard System. See the APR Supplement for more information on the application tests.



This report provides a brief performance summary using the AIM application and multiuser benchmarks. System performance will vary according to configuration, application mix and usage. More detailed analysis of these variables can be obtained with the AIM Benchmark™ Suites.

AIM Technology is the industry leader in providing benchmarking tools for UNIX systems, and also provides AIM Job Scheduler™, AIM Disk Tuner™, and AIM Job Accounting™.

For Information on other AIM Performance Reports and the AIM Benchmarks, call AIM at 408-748-8649.

AIM Performance Report™

A UNIX™ Performance Summary Based on the AIM Benchmarks.

Pyramid 9815

Performance Rating: 574%
User Rating: 64

System Configuration Tested:

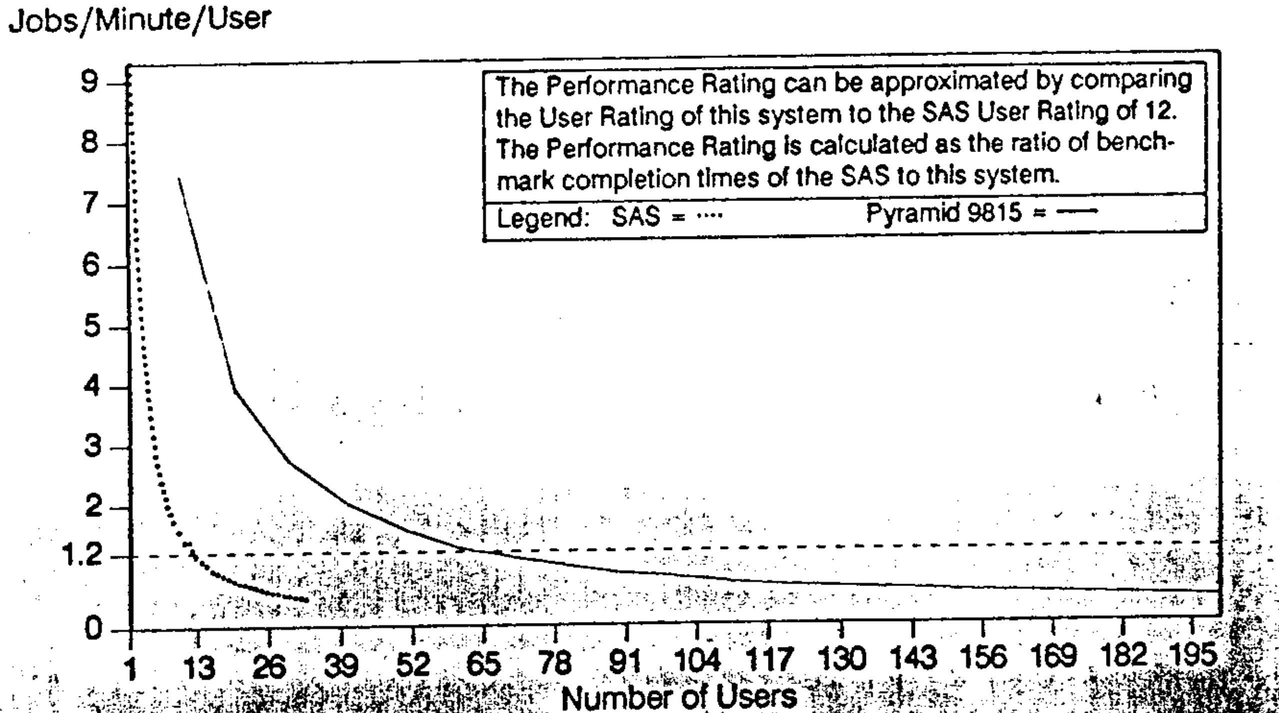
CPU	Proprietary
Clock	10Mhz
Floating Point	FPU
RAM	32Mb
Disk	1.1Gb NEC
O/S	OSx 4.4
Date Tested	May 2, 1988

The **Performance Rating** reflects the overall performance of this system, normalized to the Standard AIM System™ (SAS).† The Performance Ratings of a wide range of UNIX systems can be compared using available AIM Performance Reports.

The **User Rating** indicates the number of active users where the system's performance can become unacceptable.

Work Throughput

Work throughput as a function of the number of active users is shown below. AIM uses 1.2 jobs per minutes per user as a reference point to determine the User Rating. The actual number of users a system may accommodate will vary with the load and type of use.



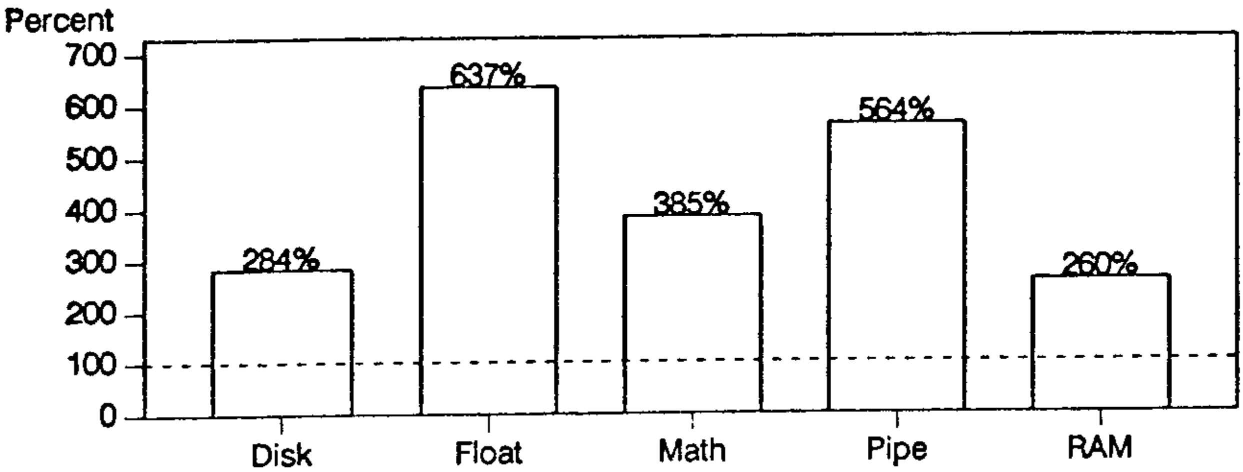
† Most VAX™ 11/780 configurations will typically rate 100% (and 12 users) of the Standard AIM System.

AIM Technology
4699 Old Ironsides Dr., Ste. 150
Santa Clara, CA 95054
408-748-8649

AIM Performance Report — Pyramid 9815

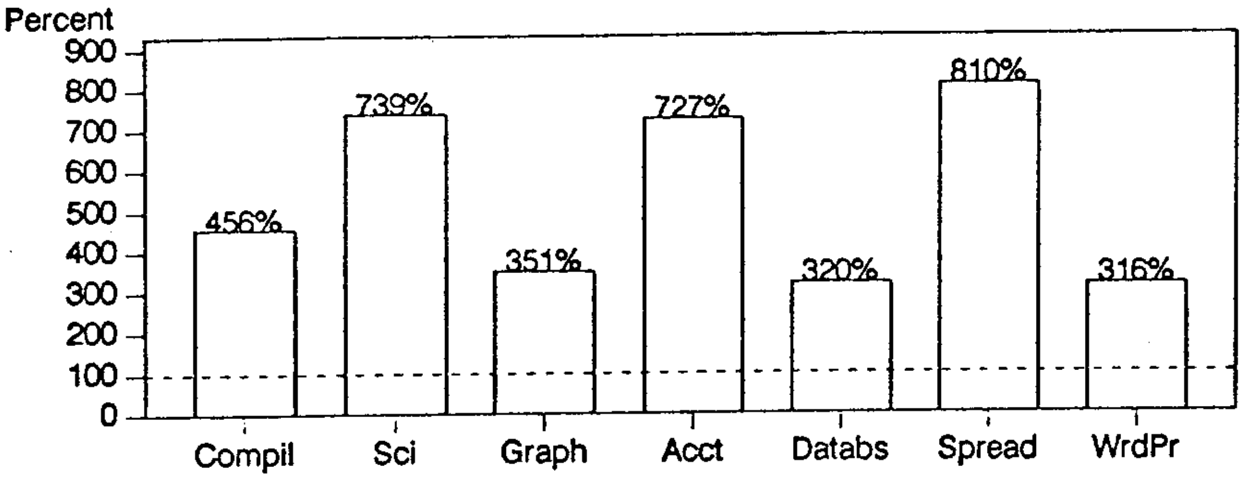
Subsystem Performance

Performance in 5 subsystem areas is shown below. Scores are normalized to the AIM Standard System. See the APR Supplement for more information on the subsystem tests.



Application Performance

Performance variations in typical engineering and business applications are shown below. Scores are normalized to the AIM Standard System. See the APR Supplement for more information on the application tests.



This report provides a brief performance summary using the AIM application and multiuser benchmarks. System performance will vary according to configuration, application mix and usage. More detailed analysis of these variables can be obtained with the AIM Benchmark™ Suites.

AIM Technology is the industry leader in providing benchmarking tools for UNIX systems, and also provides AIM Job Scheduler™, AIM Disk Tuner™, and AIM Job Accounting™.

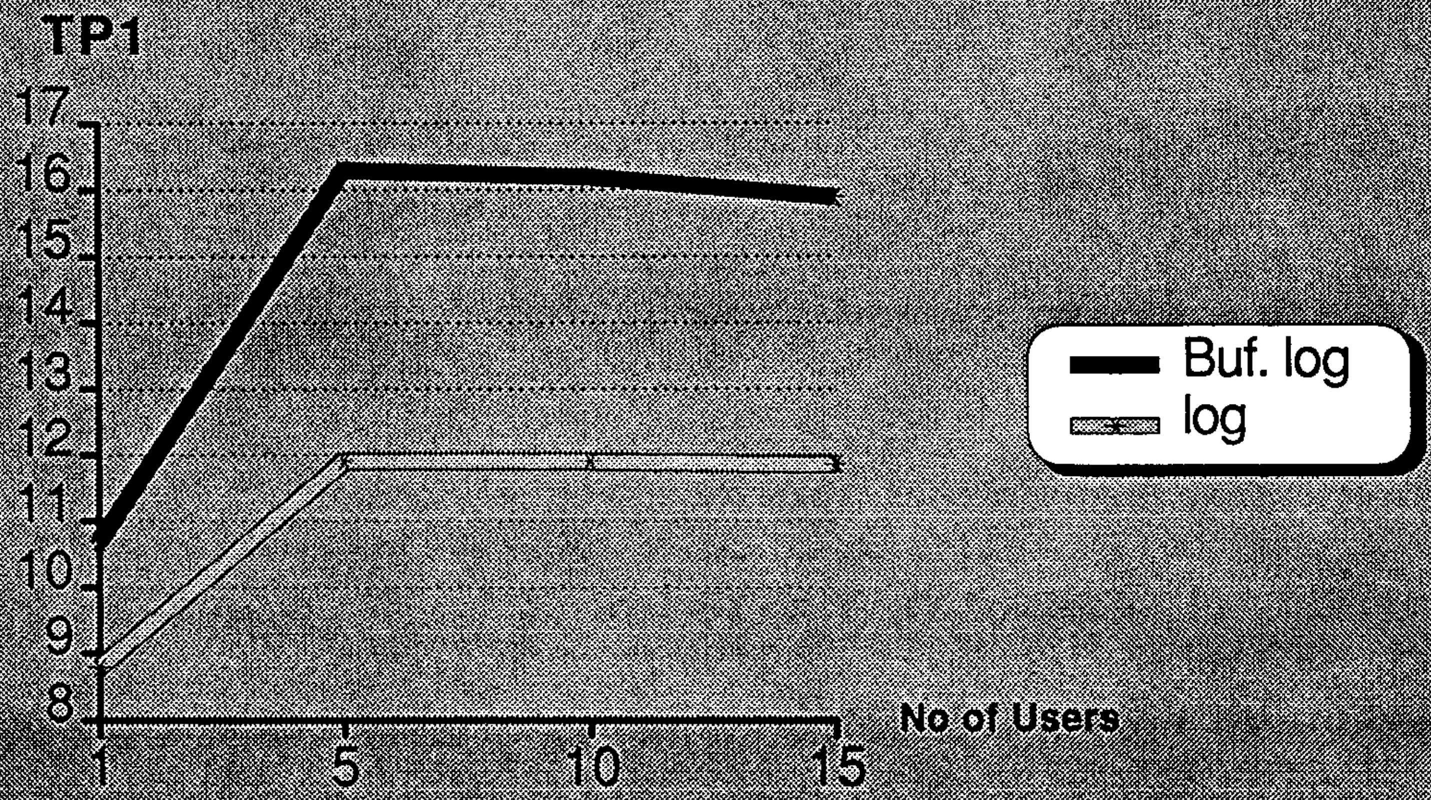
For information on other AIM Performance Reports and the AIM Benchmarks, call AIM at 408-748-8649.

I N F O R M I X - T U R B O

RC International

RC9000-30 TX 2.0.3
Informix - Turbo 1.1

TP1 Performance - TX Raw Disc I/O

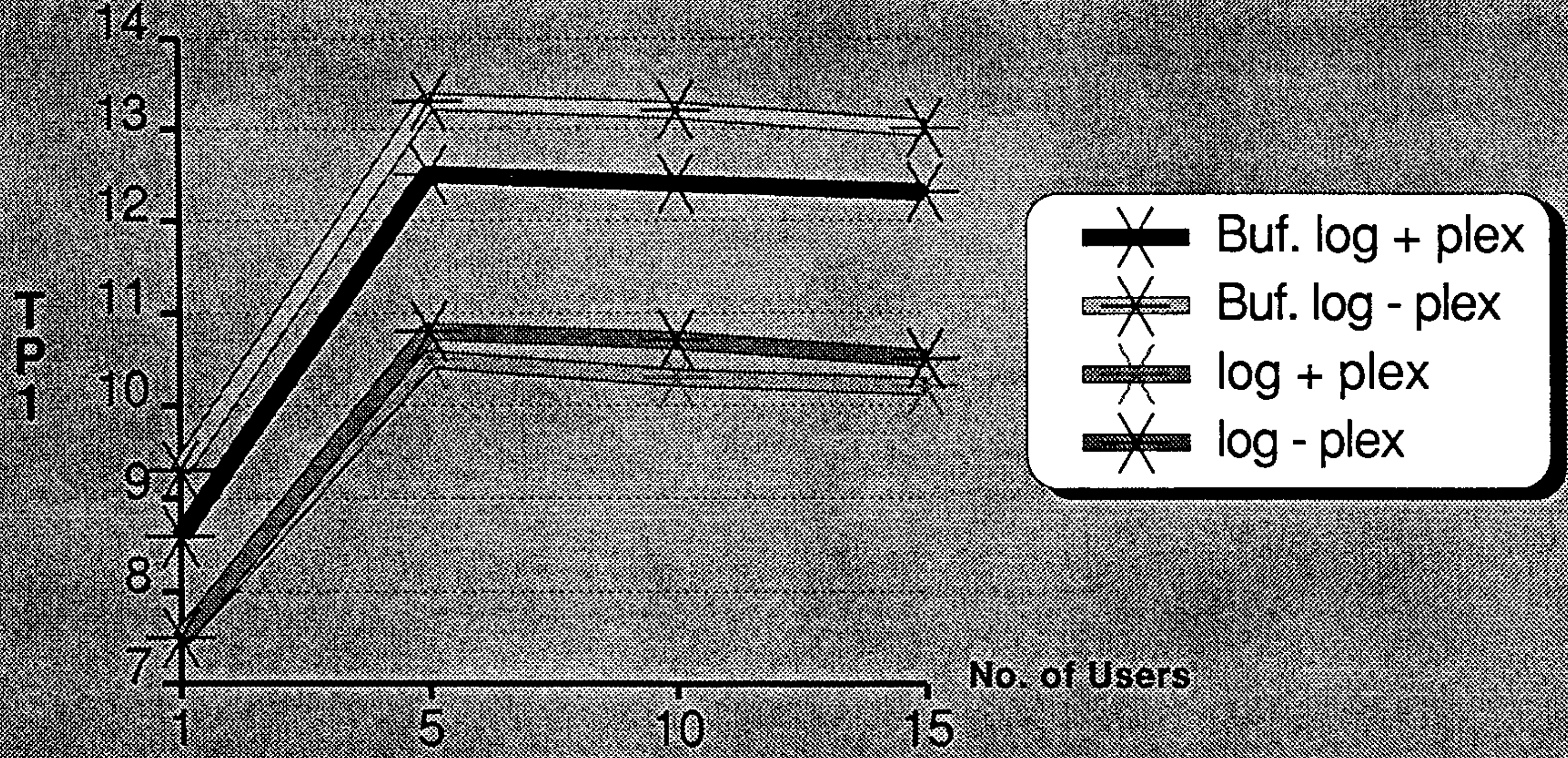


RC9000

RC International

RC9000-30 TX 2.0.3
Informix - Turbo 1.1

TP1 Performance - TX Logical Volumes



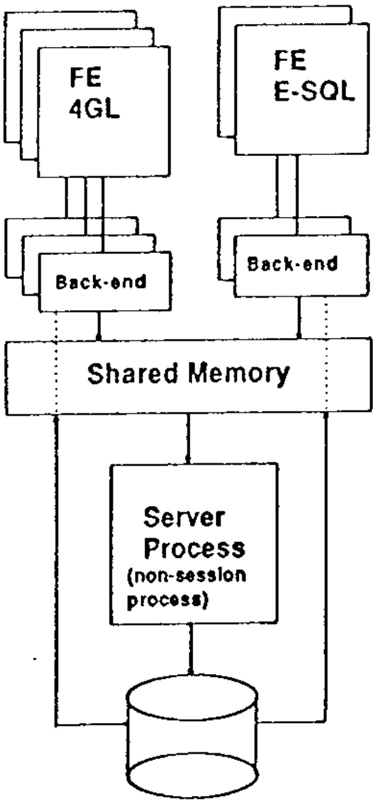
INFORMIX - TURBO 1.1

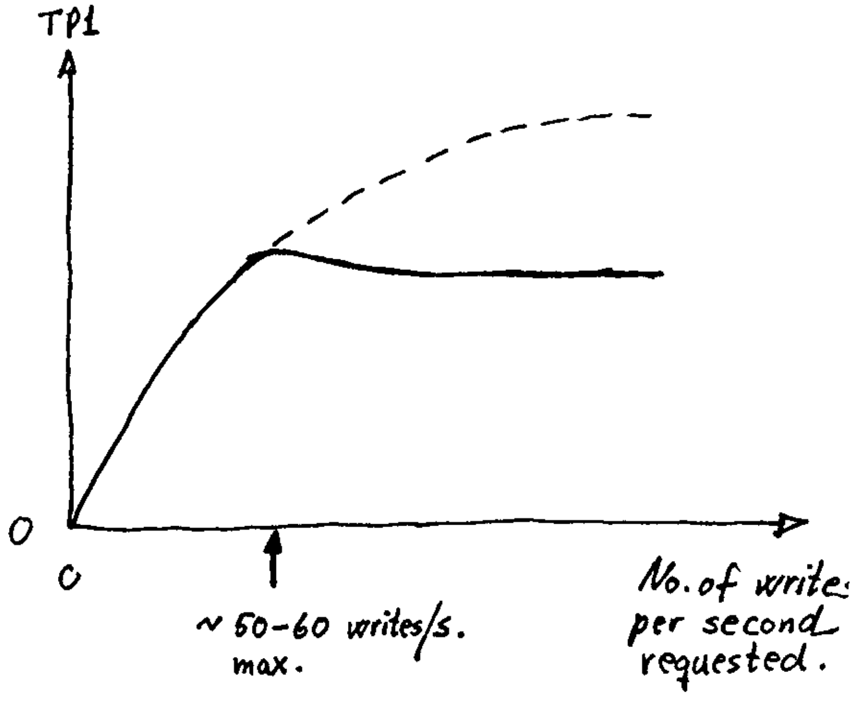
Performance Aspects

1. Structure : – Multiple Back End processes perform <u>reads</u> from discs.

   – Single Server process performs <u>writes</u> to discs.

2. Limitations : – Writes to discs are single threaded (single buffering), i.e. parallel writes to several spindles not possible.
   This limitation is removed in next Informix release (Informix-Online).





INFORMIX-ONLINE (Informix 4.0)

Performance enhancements compared to Turbo 1.1:

1. Group Commits
   Individual commits from several transactions are combined into a single write operation.

2. Multiple log buffers.
   Enables concurrent writes to several discs (log-files)

3. Multiple page cleaners.
   Enables concurrent writes from shared memory to several discs (data-files).

4. Spin-lock locking mechanism.
   Lock/unlock times reduced to few microseconds.
   Improved locking granularity and multiple queues.

5. Table insert optimization
   Improved disk space/bit-map algorithm.

6. Compiled transactions.
   Commonly used sequence of SQL statements can be precompiled and the result stored for later use.