Hot Chips–2

The SPEC and Perfect Club Benchmarks: Promises and Limitations

Rafael H. Saavedra-Barrera

Computer Science Division
University of California
Berkeley, CA 94720
(415) 642-9117
rafael@arpa.Berkeley.EDU

Santa Clara University
August 20, 1990

Organization

- Introduction
- The SPEC and Perfect Club benchmarks
- Are we ready to cast these benchmarks in stone?
- Why are analysis and prediction fundamental?
- Benchmarks do not always do what we think they should
- Characterizing benchmarks is not as easy as counting MFLOPS
- Summary and conclusions

What are the Main Conclusions of this Talk?

- Are the SPEC and Perfect suites a big improvement?

  Answer: yes

- Are these suite ready to be considered as standards?

  Answer: not yet

- Are these suites what users need to evaluate machines?

  Answer: no

Benchmarks are not Enough to Evaluate Different Machines

Machine Performance Needs to be Explained and Estimated

Benchmarks Need to be Characterized and Improved

The SPEC Benchmarks

- An effort from machine manufacturers
- A suite of 10 Fortran and C programs (scientific and systems)
- Performance is relative to the VAX-11/780
- Overall performance is computed using the geometric mean

$$\text{SPECmark} = \left[\prod_{i=1}^{n} \text{SPECratio}_i \right]^{1/n}$$

$$\text{SPECratio} = \frac{\text{machine execution time}}{\text{VAX–11/780 execution time}}$$

- Only baseline results are reported
- Main goal is to produce an industry standard benchmark suite

Programs in the Benchmark Suite

gcc	C	GNU C compiler
espresso	C	Boolean function minimization
spice2g6	Fortran	Analog circuit simulation and analysis
doduc	Fortran	Thermohydraulical simulation of a nuclear reactor
nasa7	Fortran	Seven floating-point intensive kernels
li	C	Lisp interpreter solving the 8-queens problem
eqntott	C	Builds a truth table from a boolean expression
matrix300	Fortran	Matrix operations (SAXPY)
fpppp	Fortran	Two electron integral derivative
tomcatv	Fortran	Mesh generation with Thompson solver

The Perfect Club Benchmarks

- A suite of 13 Fortran programs (scientific)
- Overall performance is computed using the harmonic mean

$$\text{Harmonic Mean} = \frac{n}{\sum_{i=1}^{n} \frac{1}{\text{MFLOPS}_i}}$$

$$\text{MFLOPS} = \frac{\text{FLOP count on 1 CPU of a CRAY X–MP}}{\text{CPU time in seconds} \times 10^6}$$

- Baseline and manual optimization results are reported
- Main goal is to match problems and algorithms with machines

Programs in the Benchmark Suite

ADM	Pseudospectral air pollution simulation
ARC2D	Two-dimensional fluid solver of Euler equations
BDNA	Molecular dynamic package for the simulation of nucleic acids
DYFESM	Structural dynamics benchmark (finite element)
FLO52	Transonic inviscid flow past an airfoil
MDG	Molecular dynamics for the simulation of liquid water
MG3D	Depth migration code
OCEAN	Two dimension ocean simulation
QCD	Quantum chromodynamics
SPEC77	Weather simulation
SPICE	Circuit simulation and analysis (spice2g6)
TRFD	A kernel simulating a two-electron integral transformation
TRACK	Missile tracking

SPEC and Perfect Benchmarks are an Improvement

- Suites of real, non-trivial, portable applications
- First attempts to document portability changes and optimizations
- Benchmarking as an ongoing process
- Benchmarking used to evaluate and design better systems
- More than benchmarking (metrics, standard, verification)

Performance is More than Benchmark Execution Times

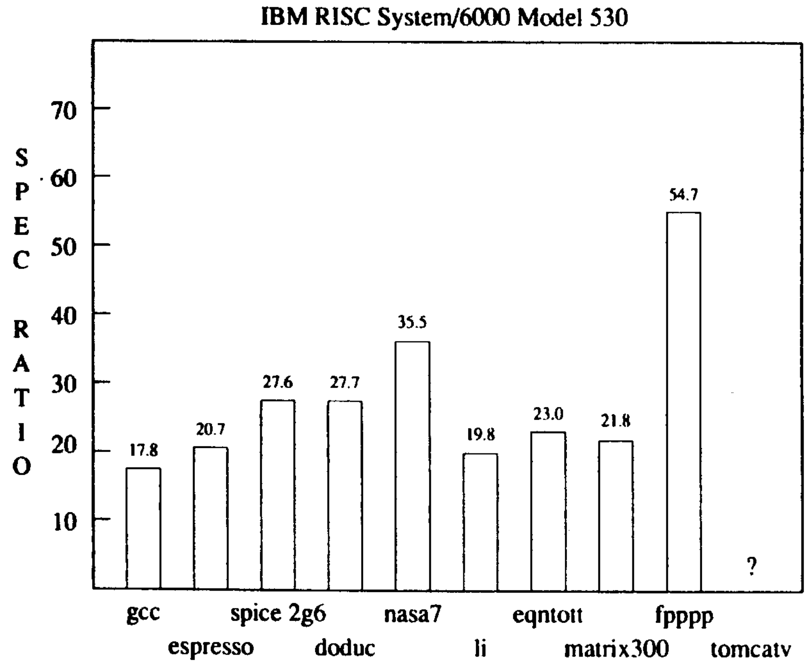
- SPECmarks, execution times and MFLOPS rates are not enough
- Users need to know what the benchmarks measure
- Users need to explain benchmark results in terms of
  - The program static and dynamic statistics
  - The machine performance characteristics
- Users need to extrapolate from benchmark results

What can we conclude from the SPEC and Perfect Club results?

Why are Analysis and Prediction Fundamental? (1/3)

Important Questions for Users

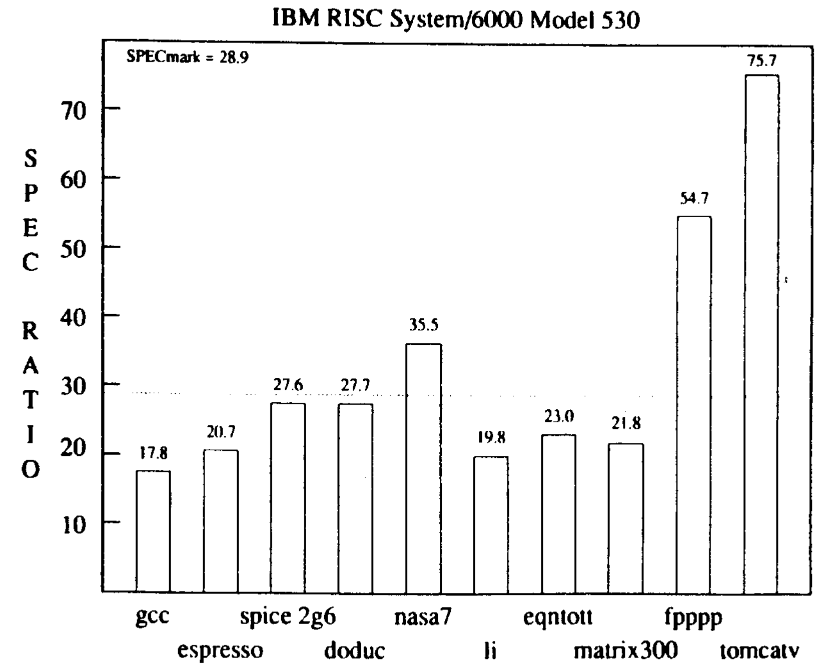
- Why is the SPECratio for fpppp so large?
- What will the SPECratio for tomcatv be?



Why Analysis and Prediction are Fundamental? (2/3)

Benchmark Results Do Not Provide the Answer

- Users can't explain from benchmark results
- Users can't extrapolate from benchmark results



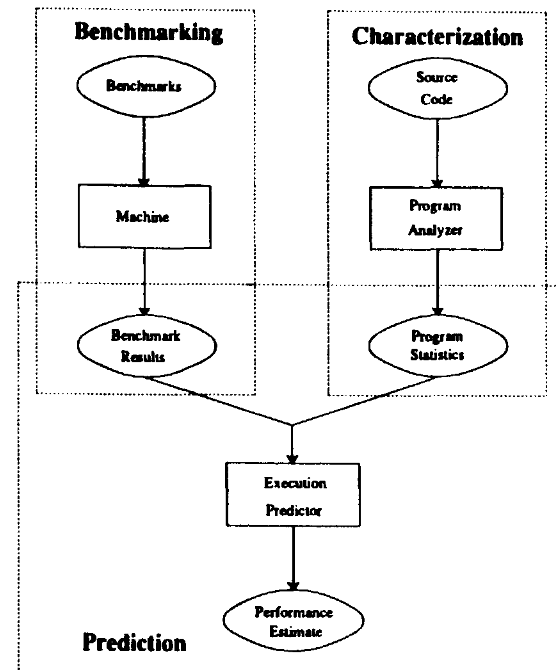
Why Analysis and Prediction are Fundamental? (3/3)

- Why is the performance of tomcatv the highest?
  - 61% of all operations are inside a single loop
  - Most of the operations are of the form multiply/add
  - 60% of all loads and stores can be eliminated (registers)
  - Optimizer reschedules instructions to avoid conflicts

- Why is the performance of fpppp high?
  - 80% of all operations are of the form multiply/add
  - The size of the basic blocks is very large (400 statements)
  - Not as much reuse of registers as in tomcatv

Benchmark Results Should Provide Insight; Not Only Numbers

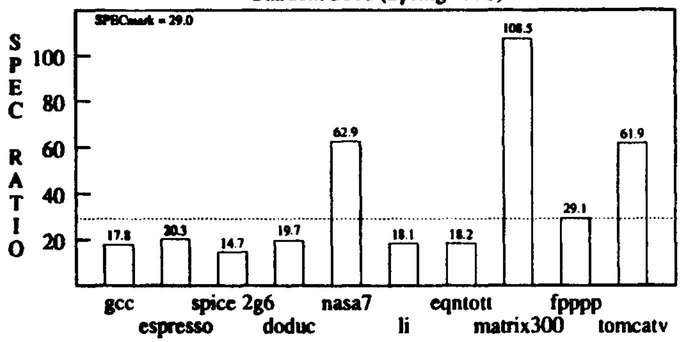
Benchmarking, Characterization, and Prediction



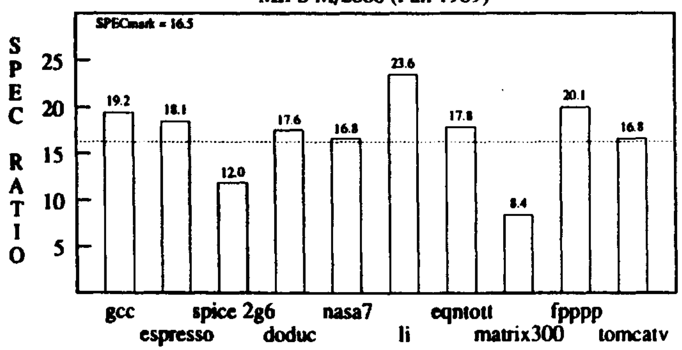
- Measure the performance of the system (benchmarking)
- Characterize the execution of programs and benchmarks
- Estimate execution time for programs on different machines
- Explain results in terms of benchmarks and machines

Benchmarking Should Answer the Interesting Questions

Stardent 3010 (Spring 1990)



MIPS M/2000 (Fall 1989)

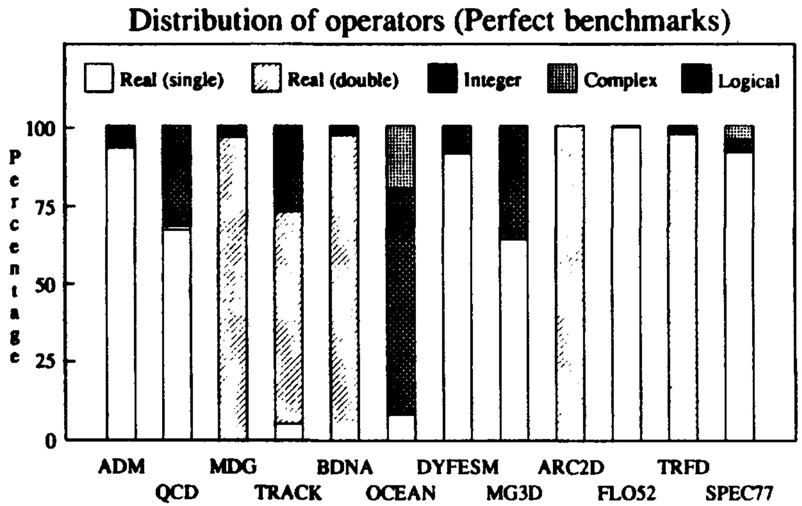


- Why does matrix300 give the highest and lowest SPECratio?
- How is matrix300 different from the other benchmarks?
- What are the main differences between the two machines?
- Why does Spice2g6 give low performance in both machines?

Dynamic Statistics: SPEC and Perfect Benchmarks

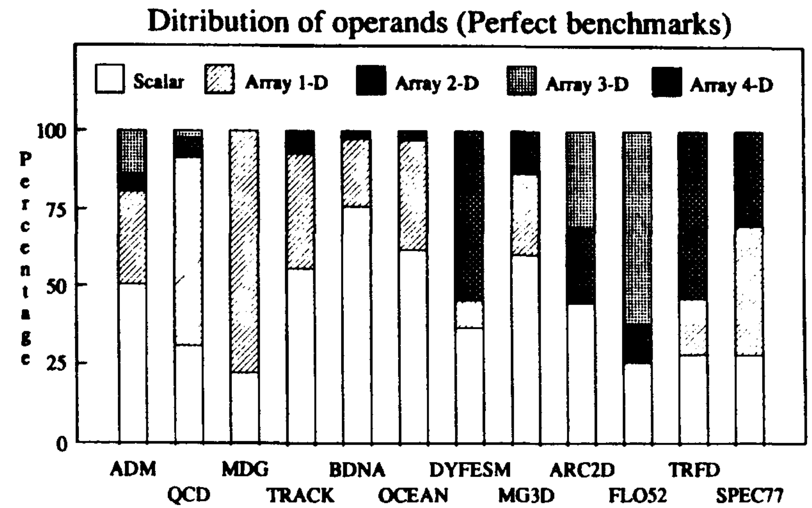
- Machine-independent statistics for an 'abstract' machine
- Only Fortran programs have been characterized
- Programs were not optimized (baseline execution)
- Statistics for Spice2g6 for 7 test cases (models)
- Model 'perfect' used by the Perfect benchmarks
- Model 'greycode' used by the SPEC benchmarks

Dynamic Statistics for the Perfect Benchmarks (1/3)



- Real (Single): ADM, DYFESM, FLO52, and SPEC77
- Real (Double): MDG, BDNA, ARC2D, and TRFD
- Real (Single) and Integer: QCD, and MG3D
- Real (Double) and Integer: TRACK
- Integer and Complex: OCEAN

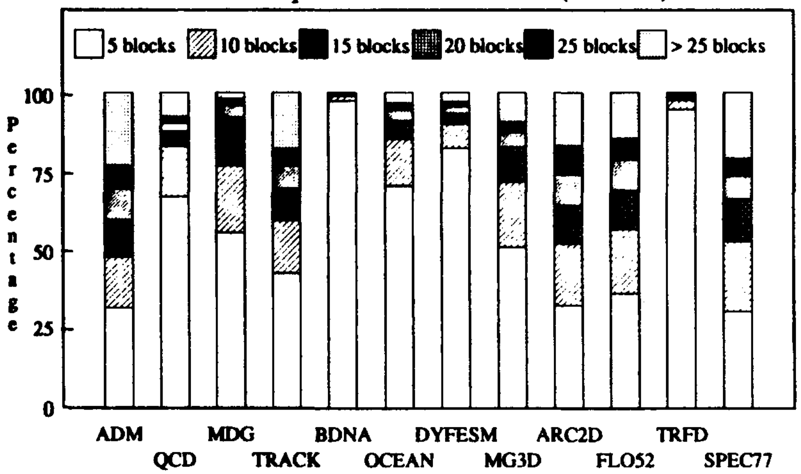
Dynamic Statistics for the Perfect Benchmarks (2/3)



- Scalar (> 50%): ADM, BDNA, TRACK, MG3D, and OCEAN
- 1-D Arrays (> 50%): QCD and MDG
- 2-D Arrays (> 50%): DYFESM and TRFD
- 3-D Arrays (> 50%): FLO52
- More Uniform: ARC2 and SPEC77

Dynamic Statistics for the Perfect Benchmarks (3/3)

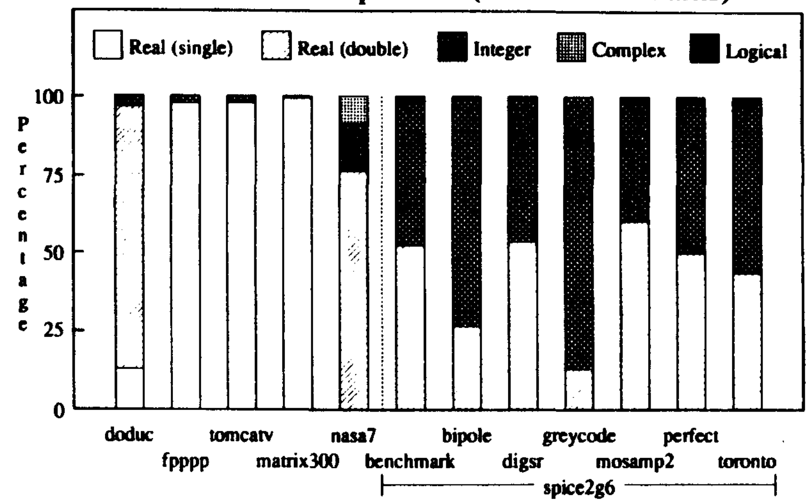
Most important basic blocks (Perfect)



- 5 blocks > 90%: BDNA and TRFD
- 5 blocks > 75%: DYFESM
- 5 blocks > 50%: QCD, MDG, MG3D, and OCEAN
- More Uniform: ADM, TRACK, ARC2D, FLO52, and SPEC77

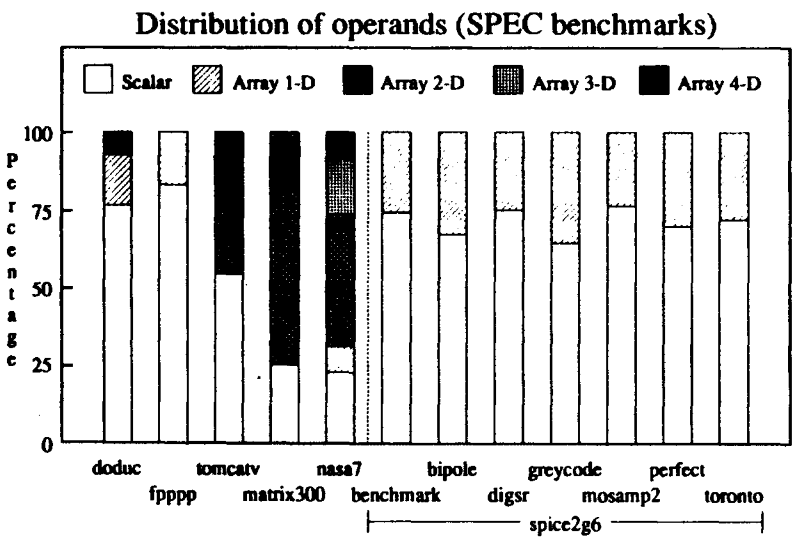
Dynamic Statistics for the SPEC Benchmarks (1/3)

Distribution of operators (SPEC benchmarks)



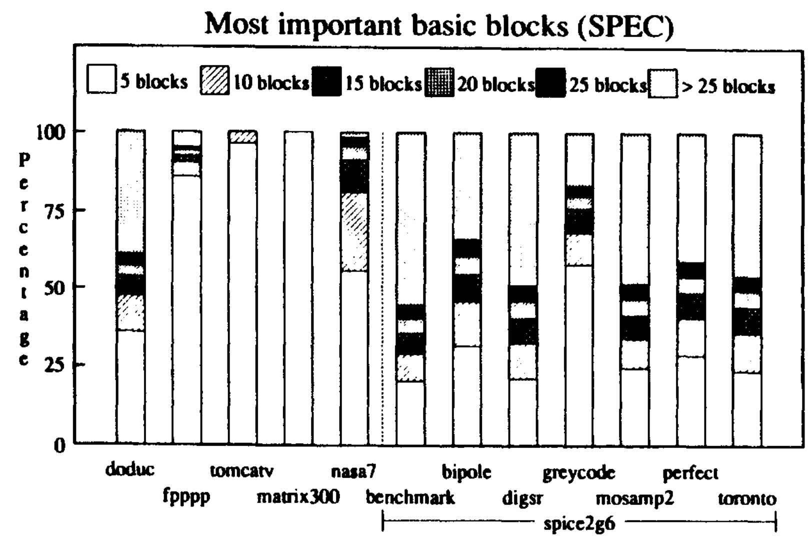
- Real (Single): fpppp, tomcatv, and matrix300
- Real (Double): doduc
- Integer and Real (Double): bipole, greycode, and toronto
- Real (Double) and Integer: benchmark, digsr, mosamp2, and perfect
- Real (Double), Integer, and Complex: nasa7

Dynamic Statistics for the SPEC Benchmarks (2/3)



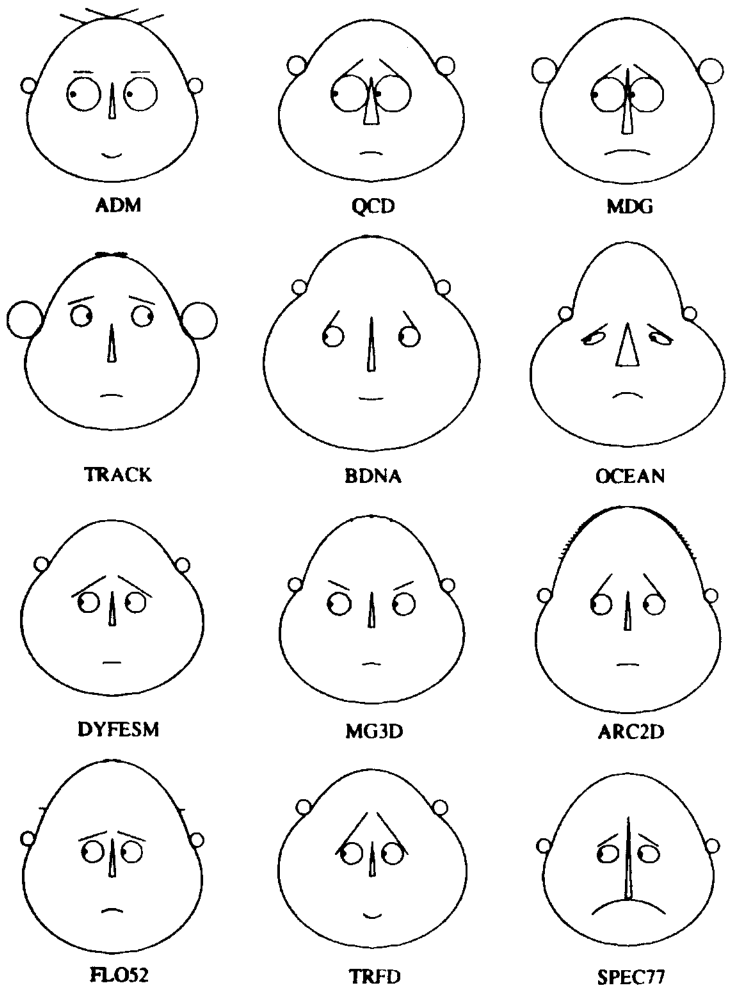
- Scalar (> 50%): doduc, fpppp, and spice2g6 (all models)
- 2-D Arrays (> 50%): matrix300
- Scalar and 2-D Arrays: tomcatv
- Scalar and 1,2,3,4-D Arrays: nasa7

Dynamic Statistics for the SPEC Benchmarks (3/3)

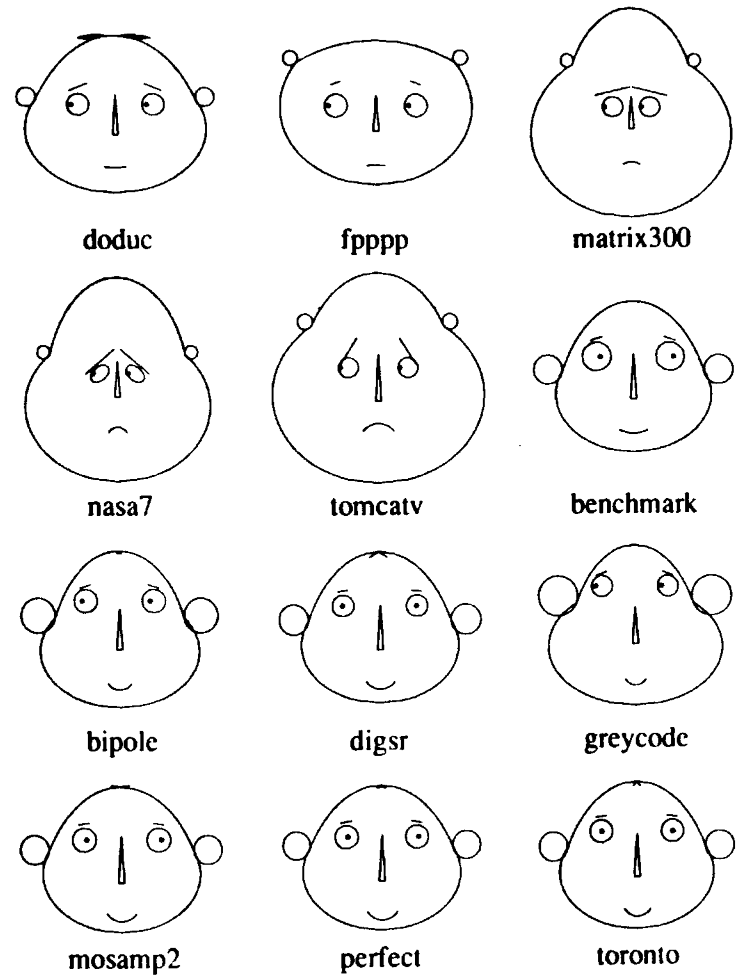


- 1 blocks > 99%: matrix300
- 5 blocks > 80%: fpppp and tomcatv
- 5 blocks > 50%: nasa7 and greycode
- More Uniform: doduc, spice2g6 (except greycode)

Chernoff Faces for the Perfect Benchmarks



Chernoff Faces for the SPEC Benchmarks



Chernoff Faces and Benchmark Similarity

- Each Chernoff face consists of 23 program characteristics
- Assignment of variables to facial features
- Random assignment of variables

Clusters for the Perfect Benchmarks:

1) FLO52, DYFESM, and TRFD
2) QCD and MDG
3) ADM and MG3D
4) ARC2D and BDNA
5) SPEC77
6) TRACK
7) OCEAN

Clusters for the SPEC Benchmarks and Spice2g6:

1) benchmark, digsr, mosamp2, perfect, and toronto
2) bipole and greycode
3) matrix300 and tomcatv
4) doduc and fpppp
5) nasa7

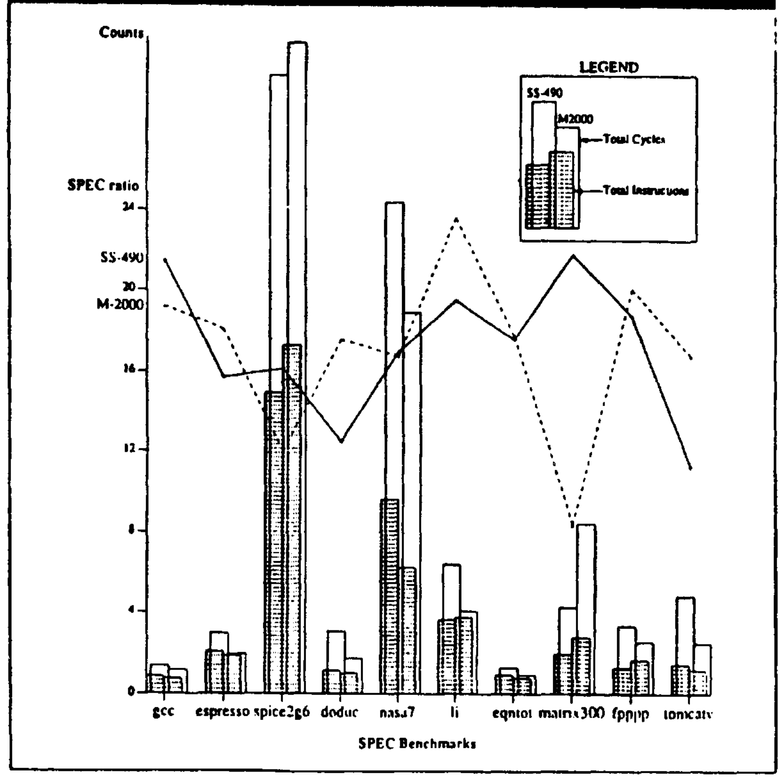
Explaining Matrix300 Results (1/3)

- Analysis of cycle counts and object code

$$Time_p = \frac{Instructions_p}{Clock \times CPI_p}$$

- Sun SPARCserver 490 (SS-490) and MIPS M/2000

Figure 1 Instruction, Cycle Count, and SPEC Ratio Comparisons of SS-490 and M/2000



Explaining Matrix300 Results (2/3)

- 99% of the operations executed are in one basic block
- Index calculation represents 43% of all operations

```
      SUBROUTINE SAXPY(N, A, X, INCX, Y, INCY)
      IMPLICIT REAL*8(A-H,O-Z),INTEGER*4(I-N)
      DIMENSION X(INCX,N), Y(INCY,N)
      IF (N.LE.0) RETURN
      DO 10 I=1,N
          Y(1,I) = Y(1,I) + A*X(1,I)
   10 CONTINUE
      RETURN
      END
```

MIPS M/2000

operation	percent of operations	percent of execution time
floating point store	14.14%	1.28%
floating point add	14.14%	6.14%
floating point multiply	14.14%	9.96%
2-D array reference	42.64%	70.38%
loop overhead	14.25%	11.89%

- Index calculation represents 70% of the execution time
- Compiler generates better code in new release

Explaining Matrix300 Results (3/3)

SPARCstation I

operation	percent of operations	execution time
floating point store	14.14%	2.42%
floating point add	14.14%	8.84%
floating point multiply	14.14%	11.10%
2-D array reference	42.64%	57.26%
loop overhead	14.25%	19.90%

- Index calculation represents 57% of the execution time

CRAY Y–MP/8128 (scalar)

operation	percent of operations	execution time
floating point store	14.14%	2.75%
floating point add	14.14%	6.26%
floating point multiply	14.14%	16.10%
2-D array reference	42.64%	16.77%
loop overhead	14.25%	57.04%

- Index calculation represents 16% of the execution time

Results Without Explanation are not very Useful

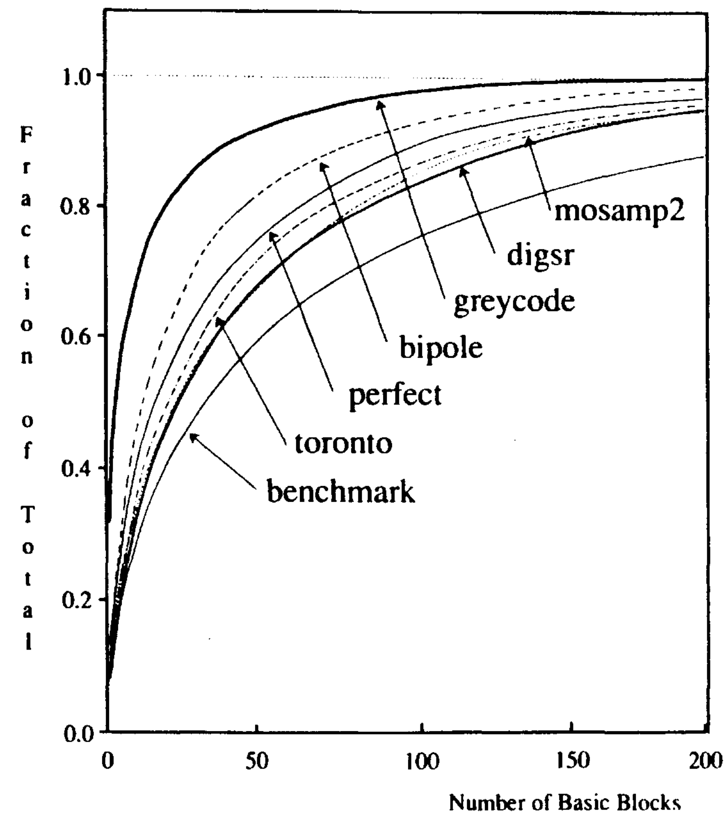
Dynamic Statistics for Spice2g6 (1/3)

Data Set	Execution Time	Basic Blocks Executed	Expressions in Assignments	Integer Operators	Double Operators
benchmark	72 s	52.5%	52.4%	46.0%	52.2%
bipole	196 s	34.9%	36.0%	72.7%	26.4%
digsr	456 s	35.4%	50.3%	45.3%	53.5%
greycode	**15289 s**	**33.3%**	**19.3%**	**86.9%**	**12.8%**
mosamp2	34 s	36.4%	55.3%	38.2%	60.1%
perfect	234 s	33.9%	52.9%	48.7%	49.7%
toronto	155 s	35.0%	46.7%	55.3%	43.3%

Data Set	Number of Basic Blocks Containing X% of All Operations			
	50%	75%	90%	95%
benchmark	33	103	232	376
bipole	13	36	84	126
digsr	25	70	143	197
greycode	3	15	43	72
mosamp2	24	69	132	202
perfect	17	51	106	155
toronto	21	59	125	182

- There are 6044 basic blocks in Spice2g6
- The SPEC benchmarks use greycode as input model
- Greycode executes longer, but touches fewer basic blocks
- More than 80% of assignments are memory to memory transfers
- Only 13% of arithmetic operations are in double precision
- 3 basic blocks execute more than 50% of all operations

Dynamic Statistics for Spice2g6 (2/3)



Greycode is not an Interesting Model for Benchmarking

Dynamic Statistics for Spice2g6 (3/3)

```
140   LOCIJ=NODPLC(IRPT+LOCIJ)
      IF (NODPLC(IROWNO+LOCIJ).EQ.I) GO TO 155
      GO TO 140
```

- This block contains 32% of all operations for greycode
- All operations in the block are between integers
- Spice2g6 is supposed to be floating point intensive

What Do the Above Results Mean?

- Large programs do not necessary make good benchmarks
- A longer execution time does not produce a better benchmark
- Users need to know what benchmarks measure
- Benchmark results should say something about the machine
- It is very difficult to construct good benchmarks

Scientific Programs Execute Not Only FLOPS (1/2)

$$\text{Instability} = \frac{\text{maximum attained MFLOPS}}{\text{minimum attained MFLOPS}}$$

	Our Measurements VAX-11/785			Perfect Club VAX-11/780
Program	MOPS	MARITH	MFLOPS	MFLOPS
ADM	0.42	0.15	0.12	0.2
ARC2D	–	–	–	0.03
BDNA	0.32	0.13	0.13	0.2
DYFESM	0.52	0.15	0.14	0.5
FLO52	0.43	0.13	0.13	0.3
MDG	0.30	0.08	0.08	0.2
MG3D	–	–	–	0.2
OCEAN	0.42	0.17	0.05	0.2
QCD	0.46	0.15	0.10	0.2
SPEC77	0.36	0.12	0.11	0.2
SPICE	0.39	0.15	0.08	0.1
TRFD	0.43	0.13	0.12	0.1
TRACK	0.35	0.13	0.10	0.1

- MOPS = millions of ‘abstract’ operations
- MARITH = millions of arithmetic operations
- MFLOPS = millions of floating point operations

Scientific Programs Execute Not Only FLOPS (2/2)

- Our Measurements

  MOPS, MARITH, MFLOPS Instabilities = 1.73, 2.13, 2.80

- Perfect Club Numbers

  Instability (with and without ARC2D) = 16.7, 5.0

What Do the Above Results Mean?

- Benchmarks execute more than just floating point operations
- We need to define a better unit of work for programs
- Characterization must be architecture independent
- Program statistics should be easy to verify

Predicting Execution Times (1/2)

Perfect Benchmarks

	ADM			QCD			MDG		
System	real (sec)	pred (sec)	error (%)	real (sec)	pred (sec)	error (%)	real (sec)	pred (sec)	error (%)
CRAY Y-MP/8128	114	98	−14.03	90	93	+3.33	4928	4511	−8.46
IBM RS/6000	208	165	−20.67	121	134	+9.70	1209	1558	+28.86
MIPS 1000	715	723	+1.11	238	328	+37.82	3026	3979	+39.49
VAX 3200	1865	1659	−11.05	1060	909	−14.24	13166	12502	−5.04
VAX-11/785	3324	2883	−13.27	2141	1701	−20.55	26401	29037	+9.98
Sun 3/50	5964	6353	+6.52	2252	2966	+31.71	29717	30273	+1.87
average			−8.56			+5.35			+11.38
r.m.s.			12.68			29.29			22.34

	TRACK			BDNA			OCEAN		
System	real (sec)	pred (sec)	error (%)	real (sec)	pred (sec)	error (%)	real (sec)	pred (sec)	error (%)
CRAY Y-MP/8128	144	139	−3.47	1357	1338	−1.42	521	524	+0.57
IBM RS/6000	–	49	–	307	288	−6.18	1025	1206	+17.65
MIPS 1000	–	115	–	–	978	–	–	2968	–
VAX 3200	337	312	+7.41	3988	3162	−20.71	8360	6434	−23.04
VAX-11/785	654	667	+1.98	6333	7446	+17.57	13651	12230	−10.41
Sun 3/50	836	994	+18.90	11986	10786	−10.01	39505	42015	+6.35
average			−1.97			−9.58			+8.02
r.m.s.			4.86			11.92			10.74

- Program statistics and machine performance measurements
- All programs executed in scalar mode
- All programs compiler with no optimization
- Missing data due to compiler errors or invalid results
- r.m.s. is the root mean square error

Predicting Execution Times(2/2)

Perfect Benchmark (cont)

	DYFESM			MG3D			ARC2D		
System	real (sec)	pred (sec)	error (%)	real (sec)	pred (sec)	error (%)	real (sec)	pred (sec)	error (%)
CRAY Y-MP/8128	131	103	-21.37	2966	2174	-26.70	3337	3025	-9.34
IBM RS/6000	-	266	-	-	6098	-	-	1516	-
MIPS 1000	651	610	-6.29	19019	15089	-20.66	-	4126	-
VAX 3200	1136	1243	+9.41	-	28850	-	-	10017	-
VAX-11/785	2059	1936	-5.97	-	50743	-	-	20082	-
Sun 3/50	4496	4986	+10.89	-	146824	-	33768	33556	-0.63
average			-2.66			-24.67			-9.34
r.m.s.			12.15			24.67			9.34

	FLO52			TRFD			SPEC77			average	r.m.s.
System	real (sec)	pred (sec)	error (%)	real (sec)	pred (sec)	error (%)	real (sec)	pred (sec)	error (%)	error (%)	error (%)
CRAY Y-MP/8128	158	136	-13.92	803	611	-23.91	516	431	-16.47	-9.18	12.65
IBM RS/6000	441	635	+43.99	403	360	-10.66	901	1241	+37.74	+15.13	29.67
MIPS 1000	1271	1406	+10.62	965	935	-3.10	-	3717	-	+2.86	19.71
VAX 3200	2822	3126	+10.77	2047	2069	+1.07	10628	11250	+5.71	-1.07	10.53
VAX-11/785	4335	4928	+13.67	3581	4153	+15.97	17846	17523	-1.81	-4.83	12.84
Sun 3/50	8024	9710	+21.01	8118	7715	-4.96	-	28616	-	+6.30	15.22
average			+14.66			-8.31			+5.85		
r.m.s.			20.26			11.84			5.85		

- Predictions for 16 machines and 30 programs
- 56% of all predictions are within 10% of execution time
- 88% of all predictions are within 20% of execution time
- 96% of all predictions are within 30% of execution time
- Prediction helps to validate performance model

Summary and Conclusions

- It is important for users to know what benchmarks measure
- Knowing what benchmarks do helps improve them
- Scientific programs do more than floating point operations
- We need to propose a realistic unit of work for programs
- Statistics are as good as the quality of the raw data
- The SPEC and Perfect suites represent a major improvement
- But more work is needed before they can become standards
- Why are academia and industry not working together?

Relevant Publications

[1] Saavedra-Barrera, R.H., "Machine Characterization and Benchmark Performance Prediction", University of California, Berkeley, Technical Report No. UCB/CSD 88/437, June 1988.

[2] Saavedra-Barrera, R.H., Smith A.J., and Miya, E. "Machine Characterization Based on an Abstract High-Level Language Machine", *IEEE Trans. on Comp.* Vol.38, No.12, December 1989, pp. 1659-1679.

[3] Saavedra-Barrera, R.H., and Smith A.J. "CPU Performance Evaluation via Benchmark Prediction", paper in preparation, University of California, Berkeley, September 1990.

Research supported in part by NASA under consortium agreement NCA2-128 and cooperative agreement NCC2-550.