Early Vector Computing

John G. Zabolitzky
www.cray-cyber.org
www.gfhr.de

High Performance Scalar

- CDC 7600 (1968) marks the limit of pipelined scalar execution
- RISC invention by Seymour Cray
- fetch, decode, execute overlapped between instructions, optimally one cycle each
- maximum performance: 1 instruction/cycle
- 36 MHz => 36 MIPS => 5 MFLOPS

Pipelined Scalar Execution

	time	====>					
instruction	1	2	3	4	5	6	7
1	fetch	decode	execute				
2		fetch	decode	execute			
3			fetch	decode	execute		
4				fetch	decode	execute	
5					fetch	decode	execute
6						fetch	decode
7							fetch
Pipelined execution on parallel functional units							

Scalar Code Example

- DO i=1,100 a(i)=b(i)*c(i)
  - load b, inc addesss
  - load c, inc address
  - multiply
  - store a, inc address
  - decrement count, loop?
- 5 instructions = cycles (optimum) for one multiply
- pipelined multiply: could start one multiply each and every cycle => only 20% efficient use

Computer Hardware Real Estate

- Memory 50%
- CPU 50%
  - Multiplier 50% (64 bit ~ 4096 elements)
  - all other 50%
- Goal: keep multiplier busy all the time
  - most cost-effective computer
  - most reliable (least components) at same performance

Architectural Alternatives

- Pipelined Scalar (RISC) as outlined
- Pipelined Vector (this presentation)
- SIMD (Single Instruction Multiple Data) parallel arithmetic (e.g., ILLIAC IV)
- Superscalar = multiple issue in one cycle
  - all modern single-chip CPUs (Intel to TI)
- VLIW (Very Long Instruction Word) variant of superscalar
- MIMD true parallel streams, e.g. Cray T3E

Alternatives Evaluation

- SIMD: too many parts => unreliable, too expensive, low usage percentage
- Superscalar: possible, but difficult c/o conflict resolution, more general than vector
- VLIW: essentially unprogrammable, but lowest hardware cost/performance
- MIMD: does not address this problem, may be superimposed upon ***ANY*** CPU structure

Vector Computation

- Scientific codes have high percentage in looping over simple data structures
- DO i=1,100 a(i) = b*c(i) + d(i)
- simple logical structure ==>
- set up such that one multiply/cycle
- one instruction for entire loop
- MFLOP rate = cycle rate or multiple thereof
- specialized for scientific/engineering tasks

Vector Pipeline c(i)=a(i)*b(i)

fetch a(i++)						
	multip. 1	multip. 2	multip. 3	multip. 4	store c(i++)	
fetch b(i++)						
						time
i=1						|
2	1					|
3	2	1				|
4	3	2	1			|
5	4	3	2	1		V
6	5	4	3	2	1	
7	6	5	4	3	2	
8	7	6	5	4	3	

First Vector Computers

- Control Data Corporation (CDC) STAR-100 [STring ARray 100 MFLOPS]
  - memory-to-memory architecture
  - therefore long startup times (~n00 cycles)
  - very slow scalar unit (~2 MFLOPS)
  - overall disappointing performance
  - contracted 1967, announced 1972, delivered 1974
  - total of 4 machines, 2 Lawrence Livermore Lab
  - Thornton (CDC) and Fernbach (LLL) loose their jobs

CDC STAR-100

Photograph courtesy of Charles Babbage Institute, University of Minnesota, Minneapolis

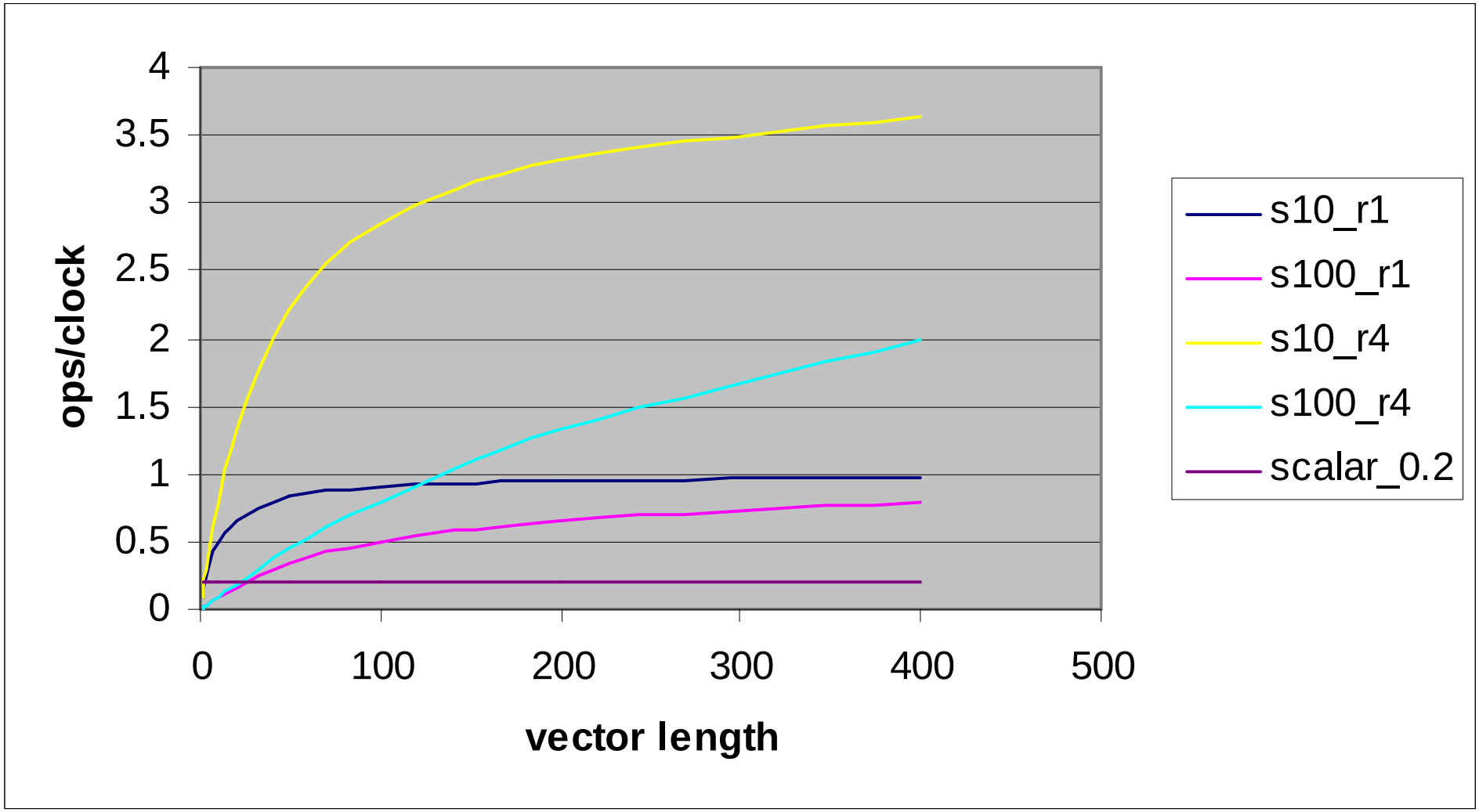
Texas Instruments ASC

- Advanced Scientific Computer
- architecturally similar to CDC STAR-100
- 7 units sold
- TI dropped out of mainframe computer manufacturing after this machine

Vector Performance I

- MFLOP rate as function of vector length
- scalar: ~constant (loop overhead)
- vector:
  - # cycles = startup + length / nflop_per_cycle
  - rate/clock = #ops / #cycles ~ n / (startup + n)
  - half rate at vectorlength n ~ startup
  - full rate needs n >> startup => “Long Vector Machine”

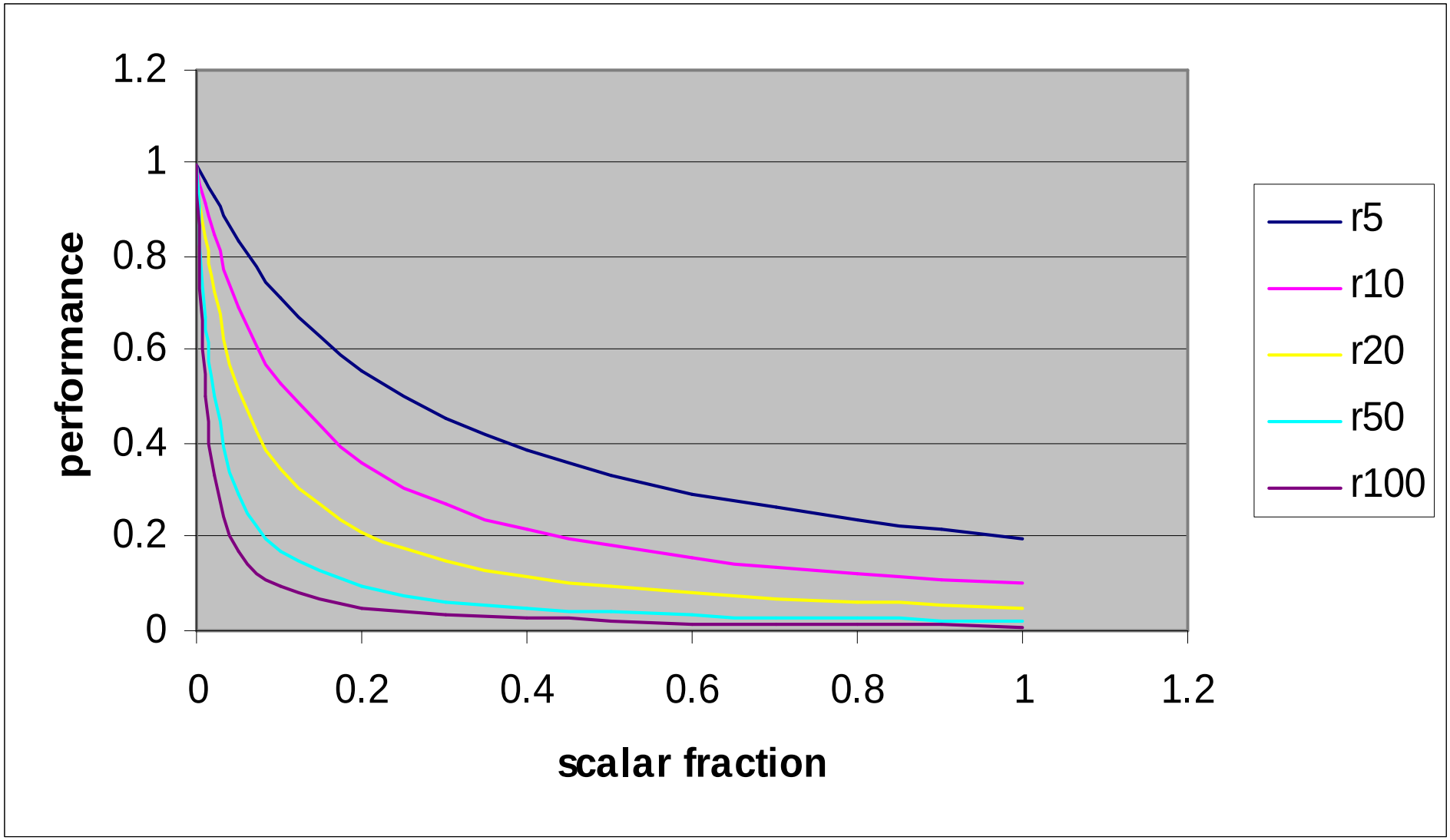
Performance vs. Startup, Length



Vector Performance II

- Vector/Scalar Subsections
  - ALL codes have some scalar (non-vectorizable) sections
  - total time = (scalar fraction)/(scalar rate) + (vector fraction)/(vector rate)
  - example: 10% / 1 MFLOPS + 90% / 100 MFLOPS = 100 / (0.1 * 100 + 0.9 * 1) = 9.2 MFLOPS !!!

Vector Version of Amdahl's Law



Vector Computer Design Guide

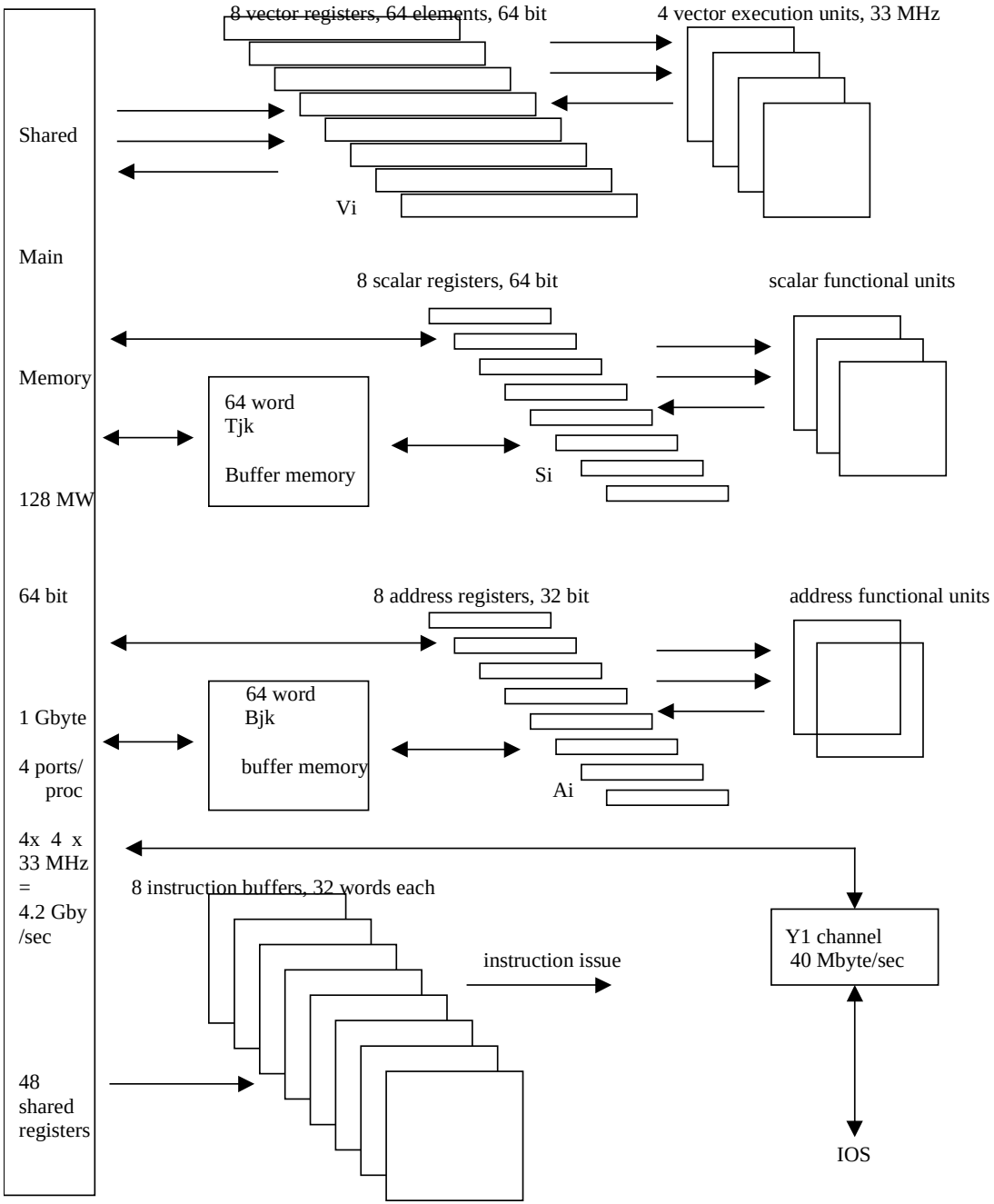
- Must have SHORT vector startup => can work with short vectors
- Must have FASTEST POSSIBLE scalar unit => can afford scalar sections
- irregular data structures ==> need gather, scatter, merge operations (and a few more)
  - x(i) = a(index(i)) * b(i)
  - y(index(i)) = c(i) + d(i)
  - where (a(i) > b(i)) c(i) = d(i)

Cray Research, Inc.

- Founded by Seymour Cray (father of CDC 6600/7600) in 1972 (STAR-100 known)
- first Cray-1 delivered in 1976 to Los Alamos Scientific Laboratory (LASL)
- 8 vector registers of 64 elements each
- Vector load/store instructions
- fastest scalar computer of its time
- 160 MFLOPS peak rate (2 ops/cycle @ 80 MHz), few cycles startup

Photograph courtesy of Charles Babbage Institute, University of Minnesota, Minneapolis

Block Diagram Cray YMP-EL, only one of four identical CPUs shown, simplified



Cray Research, Inc. cnt’d

- 1982 Cray-XMP (Chen improvements)
- 1985 Cray-2, 256 Mword memory, immersion cooled
- 1988 Cray-YMP (last Chen machine)
- 1991 Cray C90
- 1993 Cray T3D (massively parallel Alpha)
  one and only Cray-3 delivered to NCAR (Cray Comp Corp)
- 1994 Cray J90, air cooled
- 1995 Cray T3E, Cray T90 (immersion cooled)
  Cray-4 abandoned (Cray Computer Corporation ch. 11)
- 1996 acquired by Silicon Graphics
- 1998 Cray SV1 air cooled
- 1999 acquired by Teradata => Cray, Inc.
- 2002 Cray X-1, immersion spray cooled

CDC Cyber 200 Family

- 1980, enhanced version of STAR-100
- reduced startup time, ~ 50 cycles
- fast scalar unit
- rich instruction repertoire
- still memory-to-memory, 400 MFLOPS peak
- Cyber 203, Cyber 205, ETA-10 [10 GFLOPS]
- terminated in 1989 since unprofitable
- around 40 Cyber 200, 34 ETA-10 sold

Scientific Computing To-Day

- 1964-1990: all successful scientific computers have been Cray architecture machines (6600/7600/Cray-1/Cray-xyz/Japanese)
- After excursion into massively parallel slow CPUs (e.g., Cray T3E, IBM SPn) we see today the return to enhanced Cray architecture (NEC SX-6 Earth Simulator = 5104 Vector CPUs @ 8 GFLOPs air [100.000 m^3/sec] cooled; Cray-X1)
- more cost-conscious detail design