HINT: A New Way to Measure Computer Performance

John L. Gustafson and Quinn. O. Snell

In *Proceedings of the Fifth Annual Hawaii International Conference on System Sciences (HICSS)*
1995

Introduction (1 of 2)

- Early computers had single instruction stream
- Floating-point operations took longest
- Thus, computer with higher flops per second would be faster
- Wasn't linear (doubling flop/s didn't quite halve execution time) but predictions were in the "right direction"
- It doesn't work anymore…

2

Introduction (2 of 2)

- Most algorithms do more "data motion" than arithmetic
  - And "data motion" is often the bottleneck
- Growing rift in *nominal speed* (as determined by MIPS or MFLOPS) and actual *application speed*
- Using memory bandwidth figures (say, in Mbytes/sec) too simplistic
  - Each memory layer (registers, primary cache, 2nd-ary cache, main memory, disk …) has its own size and speed
  - Parallel memories make this problem worse

3

Outline

- Introduction
- Problems
- HINT
- Net QUIPS
- Examples

4

Failure of Other "Speed" Measures SPEC

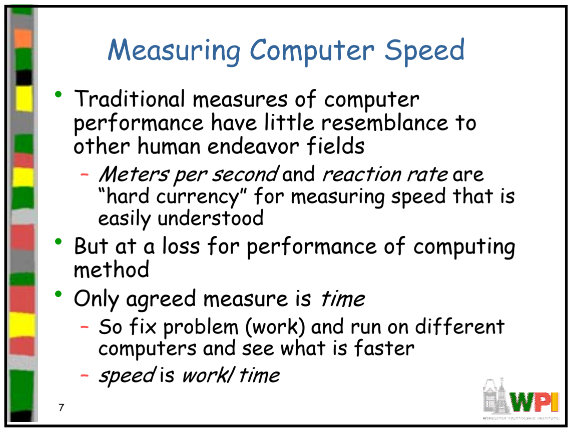
- SPEC
  - Is popular
  - Not independent (is a consortium)
  - Has to be revised when "too small" for workstations
  - Uses geometric ratio of the time reduction of various kernels
    - Compare to base machine (was VAX-11/780)
  - But some VAX-11/780 systems have SPEC mark of 3!
  - "Survives because lack of credible alternatives"

5

Failure of Other "Speed" Measures PERFECT

- PERFECT
  - Benchmark suite
  - Has 100,000 lines of (semi-) standard FORTAN
  - Not widely used since converting the application is difficult
  - Results available only for a handful of systems

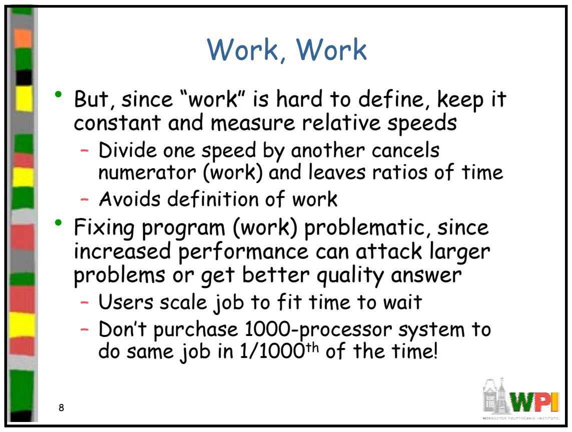
6



Measuring Computer Speed

- Traditional measures of computer performance have little resemblance to other human endeavor fields
  - *Meters per second* and *reaction rate* are "hard currency" for measuring speed that is easily understood
- But at a loss for performance of computing method
- Only agreed measure is *time*
  - So fix problem (work) and run on different computers and see what is faster
  - *speed* is *work*/ *time*

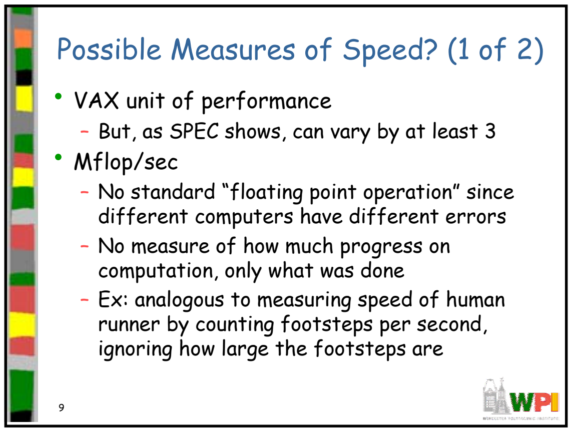
7



Work, Work

- But, since "work" is hard to define, keep it constant and measure relative speeds
  - Divide one speed by another cancels numerator (work) and leaves ratios of time
  - Avoids definition of work
- Fixing program (work) problematic, since increased performance can attack larger problems or get better quality answer
  - Users scale job to fit time to wait
  - Don't purchase 1000-processor system to do same job in 1/1000th of the time!

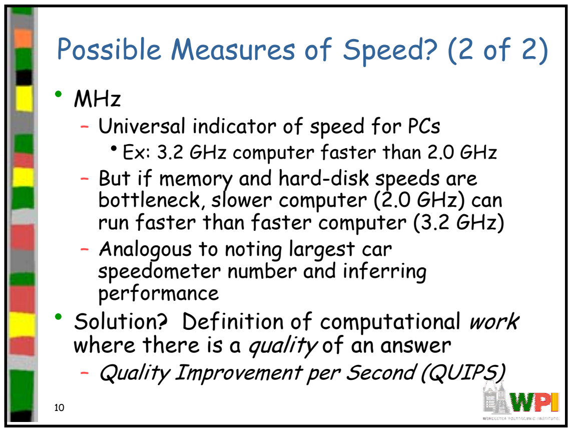
8



Possible Measures of Speed? (1 of 2)

- VAX unit of performance
  - But, as SPEC shows, can vary by at least 3
- Mflop/sec
  - No standard "floating point operation" since different computers have different errors
  - No measure of how much progress on computation, only what was done
  - Ex: analogous to measuring speed of human runner by counting footsteps per second, ignoring how large the footsteps are

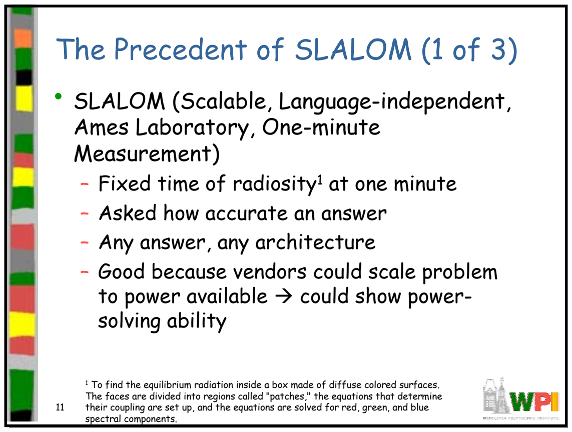
9



Possible Measures of Speed? (2 of 2)

- MHz
  - Universal indicator of speed for PCs
    - Ex: 3.2 GHz computer faster than 2.0 GHz
  - But if memory and hard-disk speeds are bottleneck, slower computer (2.0 GHz) can run faster than faster computer (3.2 GHz)
  - Analogous to noting largest car speedometer number and inferring performance
- Solution? Definition of computational *work* where there is a *quality* of an answer
  - *Quality Improvement per Second (QUIPS)*

10

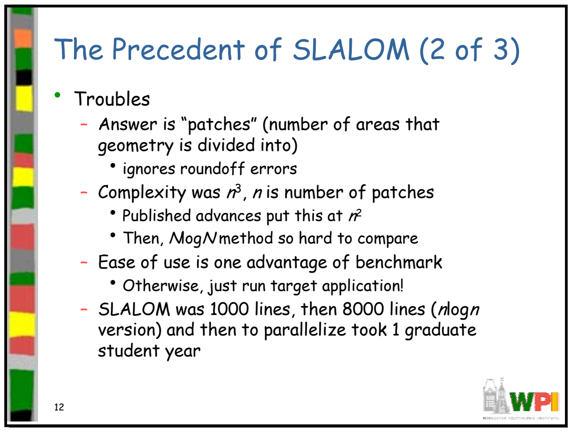


The Precedent of SLALOM (1 of 3)

- SLALOM (Scalable, Language-independent, Ames Laboratory, One-minute Measurement)
  - Fixed time of radiosity[1] at one minute
  - Asked how accurate an answer
  - Any answer, any architecture
  - Good because vendors could scale problem to power available → could show power-solving ability

[1] To find the equilibrium radiation inside a box made of diffuse colored surfaces. The faces are divided into regions called "patches," the equations that determine their coupling are set up, and the equations are solved for red, green, and blue spectral components.

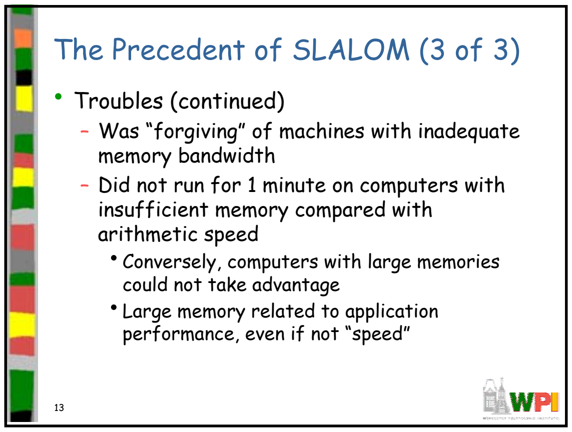
11



The Precedent of SLALOM (2 of 3)

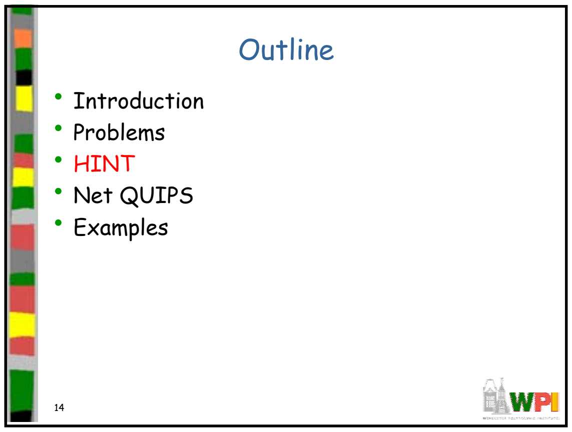
- Troubles
  - Answer is "patches" (number of areas that geometry is divided into)
    - ignores roundoff errors
  - Complexity was n^3, n is number of patches
    - Published advances put this at n^2
    - Then, *MogN* method so hard to compare
  - Ease of use is one advantage of benchmark
    - Otherwise, just run target application!
  - SLALOM was 1000 lines, then 8000 lines ($n\log n$ version) and then to parallelize took 1 graduate student year

12



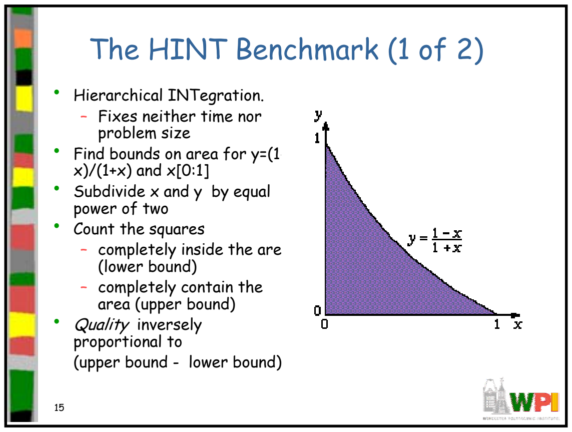
The Precedent of SLALOM (3 of 3)

- Troubles (continued)
  - Was “forgiving” of machines with inadequate memory bandwidth
  - Did not run for 1 minute on computers with insufficient memory compared with arithmetic speed
    - Conversely, computers with large memories could not take advantage
    - Large memory related to application performance, even if not “speed”



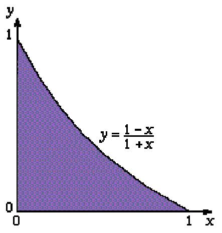
Outline

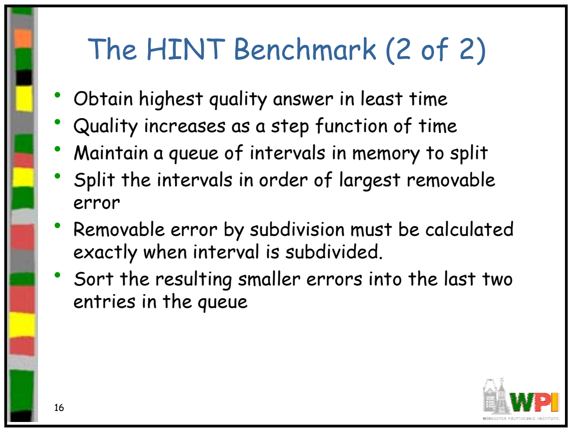
- Introduction
- Problems
- HINT
- Net QUIPS
- Examples



The HINT Benchmark (1 of 2)

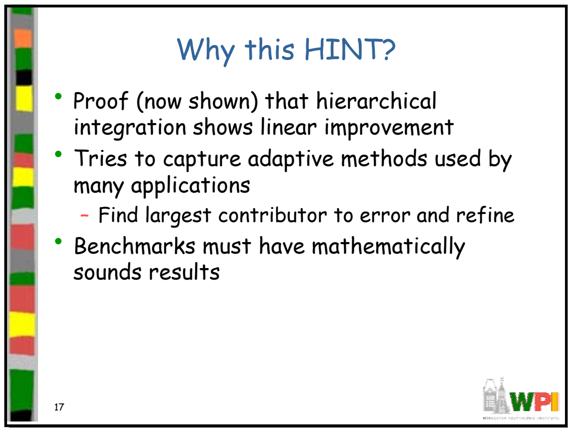
- Hierarchical INTegration.
  - Fixes neither time nor problem size
- Find bounds on area for y=(1 x)/(1+x) and x[0:1]
- Subdivide x and y by equal power of two
- Count the squares
  - completely inside the are (lower bound)
  - completely contain the area (upper bound)
- *Quality* inversely proportional to (upper bound - lower bound)





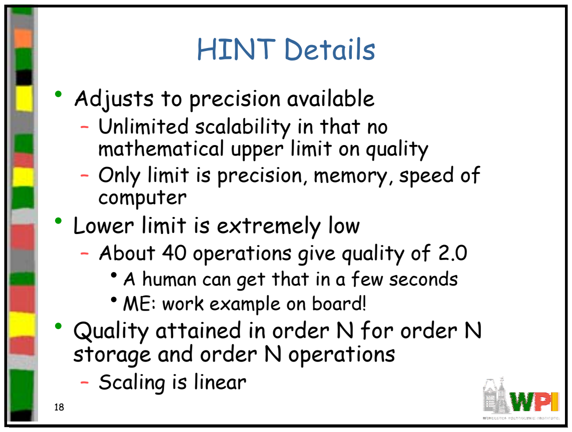
The HINT Benchmark (2 of 2)

- Obtain highest quality answer in least time
- Quality increases as a step function of time
- Maintain a queue of intervals in memory to split
- Split the intervals in order of largest removable error
- Removable error by subdivision must be calculated exactly when interval is subdivided.
- Sort the resulting smaller errors into the last two entries in the queue



Why this HINT?

- Proof (now shown) that hierarchical integration shows linear improvement
- Tries to capture adaptive methods used by many applications
  - Find largest contributor to error and refine
- Benchmarks must have mathematically sounds results



HINT Details

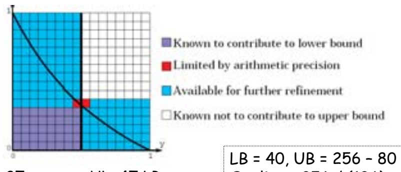
- Adjusts to precision available
  - Unlimited scalability in that no mathematical upper limit on quality
  - Only limit is precision, memory, speed of computer
- Lower limit is extremely low
  - About 40 operations give quality of 2.0
    - A human can get that in a few seconds
    - ME: work example on board!
- Quality attained in order N for order N storage and order N operations
  - Scaling is linear

HINT Example (1 of 3)

- Given word size b_d bits, x-axis represented by $b_d/2$ bits, yaxis $b_d/2$ bits
  - Ex: d = 8 bits, so x-axis [0:15], y-axis [0:15]
- If n_x and n_x are numbers of area units along x, y then
  - Compute $(1-x)/(1+x)$ as $n_y(n_x-i)/(n_x+i)$
  - Rounding up will be used for upper bound
  - Rounding down will be used for lower bound
- Then divide by n_y

HINT Example (2 of 3)

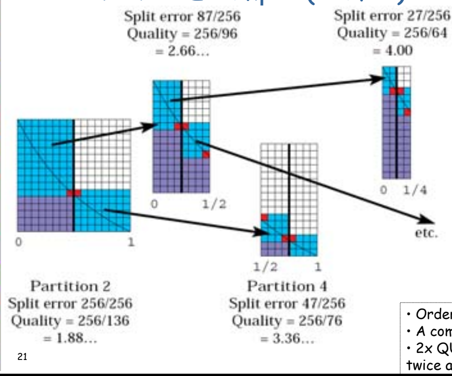
- $x = \frac{1}{2}$ then i=8, $n_x = 16$, $n_y = 16$
- $n_y(n_x-i)/(n_x+i)$
  $= 16(16-8)/(16+8) = 128/24$
  - Round down = 5, Round up = 6
- So, $5/16 < f(1/2) < 6/16$


1
0
1
y
Known to contribute to lower bound
Limited by arithmetic precision
Available for further refinement
Known not to contribute to upper bound

- 87 squares UL, 47 LR
- Should next sub-divide 87

LB = 40, UB = 256 – 80
Quality = 256 / (136)
= 1.88

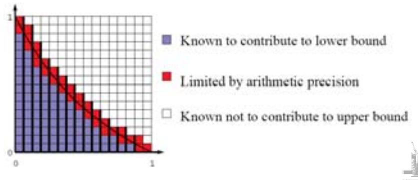
HINT Example (3 of 3)



- Order N
- A computer with
- 2x QUIPS is twice as powerful

Termination

- If no loss in precision, quality then related to number of partitions
- When width is one square or UB – LB < 2 squares then done → "insufficient precision"

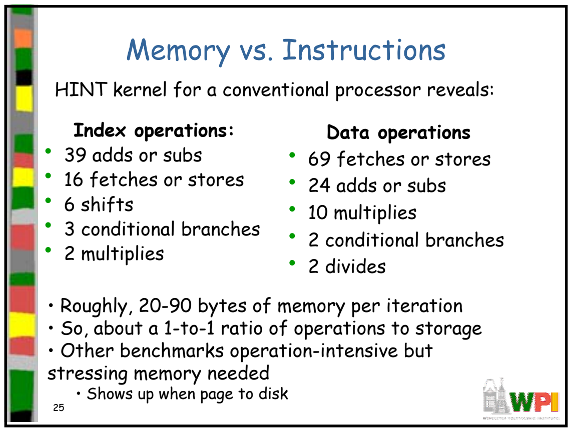

1
0
0
1
Known to contribute to lower bound
Limited by arithmetic precision
Known not to contribute to upper bound

Memory Requirements

- Must compute and store record of upper-lower bounding rectangle for each region
  - Left and right x values x_l, xr
  - UB and LB
- If b_d bits for data and b_i bits for index
  - n iterations is $(9b_d + 4b_i)n$ bits
- Note, program storage varies widely but should not be bottleneck
  - If want to stress instruction caching, do not use HINT

Data Types

- Can use floating points instead of integers
  - Roughly, 40 FLOPs per HINT iteration
- Computers have roughly same QUIPS for different data types
  - But specialized may do better.
    - Ex: scientific may have better QUIPS for floating point while business may have better QUIPS for integer



Memory vs. Instructions

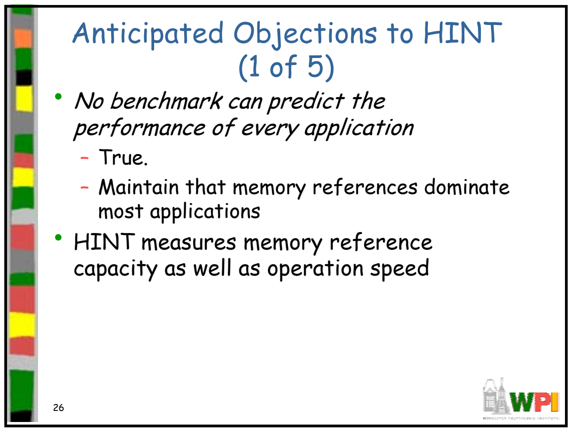
HINT kernel for a conventional processor reveals:

Index operations:
- 39 adds or subs
- 16 fetches or stores
- 6 shifts
- 3 conditional branches
- 2 multiplies

Data operations
- 69 fetches or stores
- 24 adds or subs
- 10 multiplies
- 2 conditional branches
- 2 divides

- Roughly, 20-90 bytes of memory per iteration
- So, about a 1-to-1 ratio of operations to storage
- Other benchmarks operation-intensive but stressing memory needed
  - Shows up when page to disk

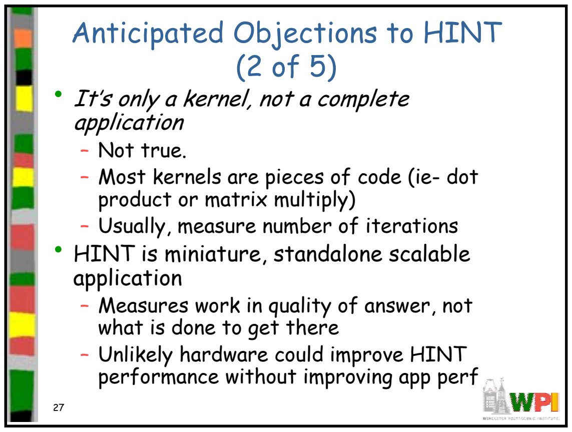
25



Anticipated Objections to HINT (1 of 5)

- *No benchmark can predict the performance of every application*
  - True.
  - Maintain that memory references dominate most applications
- HINT measures memory reference capacity as well as operation speed

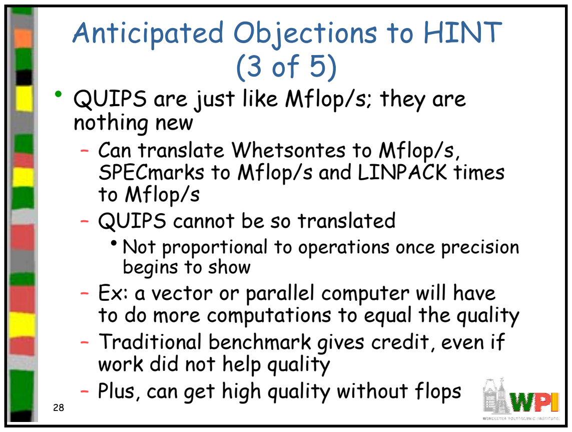
26



Anticipated Objections to HINT (2 of 5)

- *It's only a kernel, not a complete application*
  - Not true.
  - Most kernels are pieces of code (ie- dot product or matrix multiply)
  - Usually, measure number of iterations
- HINT is miniature, standalone scalable application
  - Measures work in quality of answer, not what is done to get there
  - Unlikely hardware could improve HINT performance without improving app perf

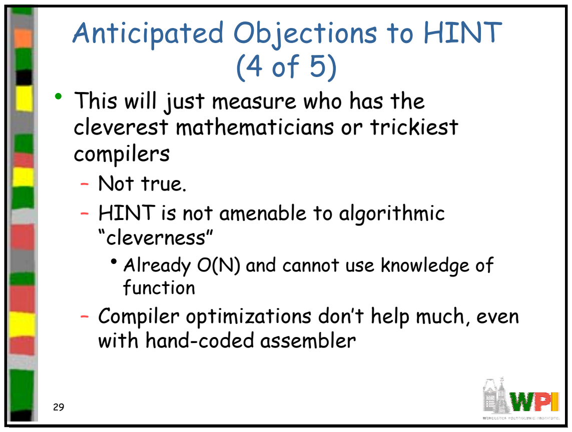
27



Anticipated Objections to HINT (3 of 5)

- QUIPS are just like Mflop/s; they are nothing new
  - Can translate Whetsontes to Mflop/s, SPECmarks to Mflop/s and LINPACK times to Mflop/s
  - QUIPS cannot be so translated
    - Not proportional to operations once precision begins to show
  - Ex: a vector or parallel computer will have to do more computations to equal the quality
  - Traditional benchmark gives credit, even if work did not help quality
  - Plus, can get high quality without flops

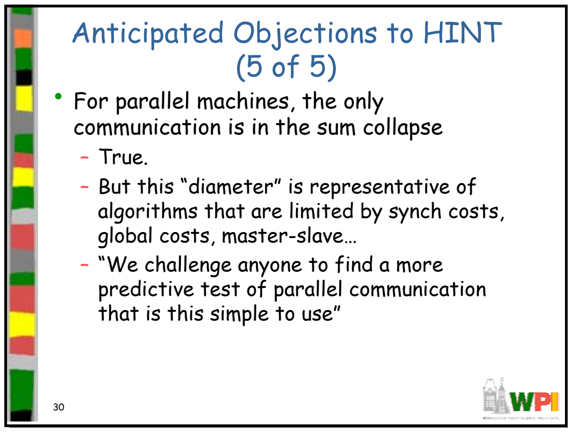
28



Anticipated Objections to HINT (4 of 5)

- This will just measure who has the cleverest mathematicians or trickiest compilers
  - Not true.
  - HINT is not amenable to algorithmic "cleverness"
    - Already O(N) and cannot use knowledge of function
  - Compiler optimizations don't help much, even with hand-coded assembler

29



Anticipated Objections to HINT (5 of 5)

- For parallel machines, the only communication is in the sum collapse
  - True.
  - But this "diameter" is representative of algorithms that are limited by synch costs, global costs, master-slave…
  - "We challenge anyone to find a more predictive test of parallel communication that is this simple to use"

30

Outline

- Introduction
- Problems
- HINT
- Net QUIPS
- Examples

31

Single Number Rating

- Tug-of-War between distributors of data and interpreters of data
  - Distributors produce lots of data showing different facets of measurements
  - Interpreters want one number to answer "How good is it?"
- So, QUIPS vs. time or QUIPS vs. mem will be distilled
- Have devised a method
  - → Net QUIPS

32

Net QUIPS (1 of 3)

- Integral of the quality (Q) divided by time^2, from time of first improvement (t_0) to last time measured

$$\text{Net QUIPS} = \int_{\log}(t_0)\text{QUIPS}(t)\ d(\log t)$$

$$= \int_{\log}(t_0) Q(t) / t\ d(\log t) = \int t_0 Q(t) / t^2\ dt$$

- Same as area under QUIPS curve on log(time) scale
- Net QUIPS units are still QUality Improvements Per Second

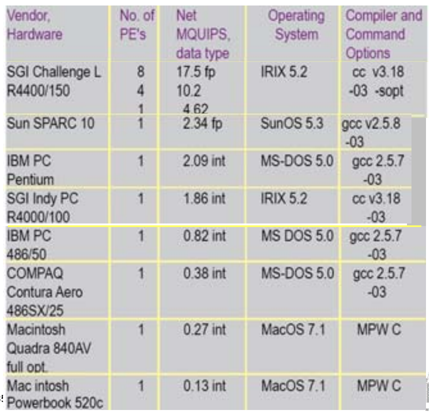
33

Net QUIPS (2 of 3)

- More memory or more cache, then QUIPS high for larger range of time
  - Net QUIPS higher
- Improved precision lifts overall Q
  - Net QUIPS higher
- Lack of interruptions (say, OS)
  - Net QUIPS higher
- Philosophically, Net QUIPS totals QUIPS weighted inversely with time to get there

34

Net QUIPS Examples



Vendor, Hardware	No. of PE's	Net MQUIPS, data type	Operating System	Compiler and Command Options
SGI Challenge L R4400/150	8 4 1	17.5 fp 10.2 4.62	IRIX 5.2	cc v3.18 -03 -sopt
Sun SPARC 10	1	2.34 fp	SunOS 5.3	gcc v2.5.8 -03
IBM PC Pentium	1	2.09 int	MS-DOS 5.0	gcc 2.5.7 -03
SGI Indy PC R4000/100	1	1.86 int	IRIX 5.2	cc v3.18 -03
IBM PC 486/50	1	0.82 int	MS DOS 5.0	gcc 2.5.7 -03
COMPAQ Contura Aero 486SX/25	1	0.38 int	MS-DOS 5.0	gcc 2.5.7 -03
Macintosh Quadra 840AV full opt.	1	0.27 int	MacOS 7.1	MPW C
Mac intosh Powerbook 520c	1	0.13 int	MacOS 7.1	MPW C

35

Net QUIPS (3 of 3)

- Hopefully, users can interpret QUIPS versus time and not use Net QUIPS
- Can be used to make "speedup" plots for multiprocessors
  - Shows not quite linear with number of processors, which is common in practice
- Can be used for humans, too
  - College-educated adults have about 0.1 QUIPS
  - Humans increase precision dynamically as needed

36

HINT Claypool (1 of 2)

- Download source code
  - cs.wpi.edu, Linux cs 2.4.25

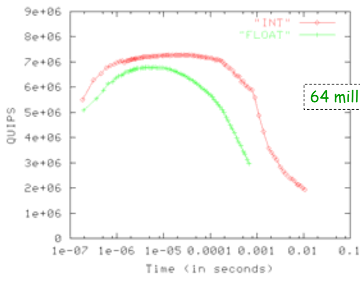
```
claypool 108 cs=>>wc -l hint.c hint.h
      343 hint.c
      170 hint.h
      513 total
```

- Compiled "out of the box" (make)
- Make "data" dir (`mkdir data`)
- Run run.sh (`sh run.sh`) or (`perl run.pl`)
- Plot 1st two columns, logscale xaxis

```
gnuplot
> set logscale x
> Plot "INT" with linesp, "FLOAT" with linesp
```

37

HINT Claypool (2 of 2)



64 million Net QUIPs

```
OS          : Linux 2.4.25
model name  : AMD Athlon(tm)
stepping    : 2
```

```
cpu MHz       : 1190
cache size    : 256 KB
MemTotal      : 1550448 KB
```

38

Extra Credit for Next Class

- Run HINT on machine of your choice
  - Download code from `http://hint.byu.edu/pub/HINT/source/`
- QUIPS Graph (ala previous slides)
  - INT, FLOAT or other …
- Report
  - Net QUIPS (returned by software)
  - CPU, OS, Memory
- Email to me and we'll discuss, build a modern Net QUIPS table

39

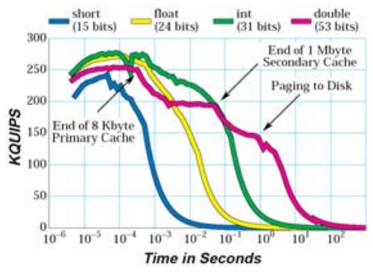
Outline

- Introduction
- Problems
- HINT
- Net QUIPS
- Examples

40

Examples – SGI Indy SC

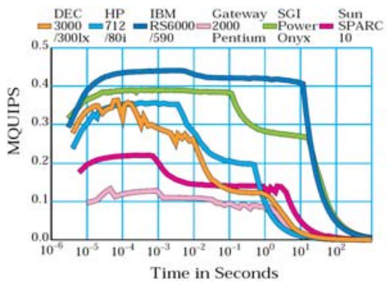
•Double, float, int, short = 53 bits, 24 bits, 32 bits, 15 bits of precision



• Using memory as x-axis is how see dropoff at caches

41

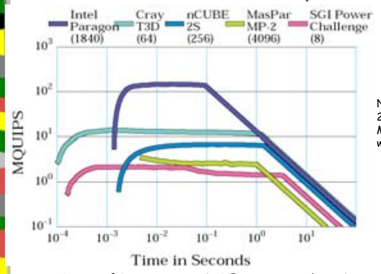
Other Workstations



- SPEC benchmark correlates with 10^{-3} and 10^{-2}
- Fits in cache of many computers

42

Parallel Computers



Note Intel Mflops is 25x the nCUBE → Nonsense! Memory bwidth is about 2x, which is captured by HINT

- Ratio of Paragon to nCUBE correspond to observed app performance
- Ratio per processor is consistent with NAS benchmark
- But
  - NAS benchmark takes 4 months to port and tune
  - HINT takes about 2 hours

43

Conclusions

- HINT is designed to last
- Fair comparisons over extreme variations in computer arch, storage capacity, precision
- Linear in answer quality, memory usage and operations
- Low cost to convert
- Speed measure that is as pure and "information-theoretic" as possible, yet practical and useful predictor of app performance

44