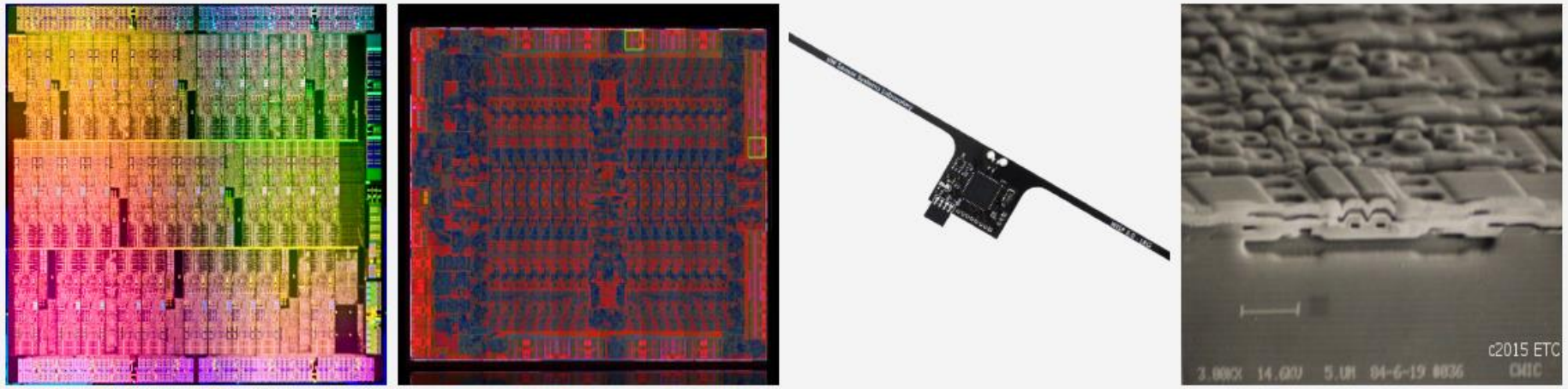
18-344: Computer Systems and the Hardware-Software Interface

Fall 2023



Course Description

Lecture 9: Introduction to Performance Evaluation

This course covers the design and implementation of computer systems from the perspective of the hardware software interface. The purpose of this course is for students to understand the relationship between the operating system, software, and computer architecture. Students that complete the course will have learned operating system fundamentals, computer architecture fundamentals, compilation to hardware abstractions, and how software actually executes from the perspective of the hardware software/boundary. The course will focus especially on understanding the relationships between software and hardware, and how those relationships influence the design of a computer system's software and hardware. The course will convey these topics through a series of practical, implementation-oriented lab assignments.

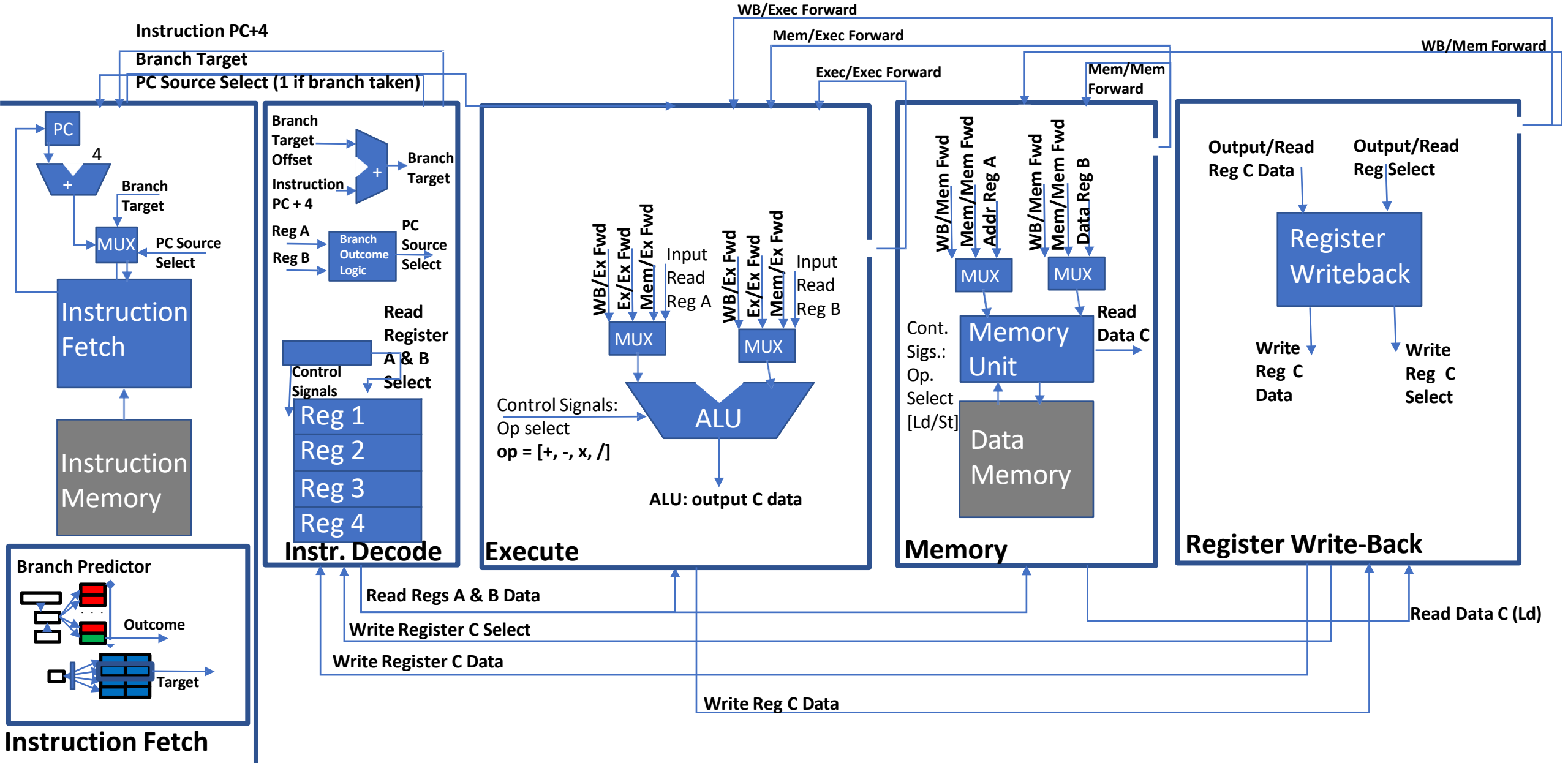
Credit: Brandon Lucia

Today: Performance Evaluation

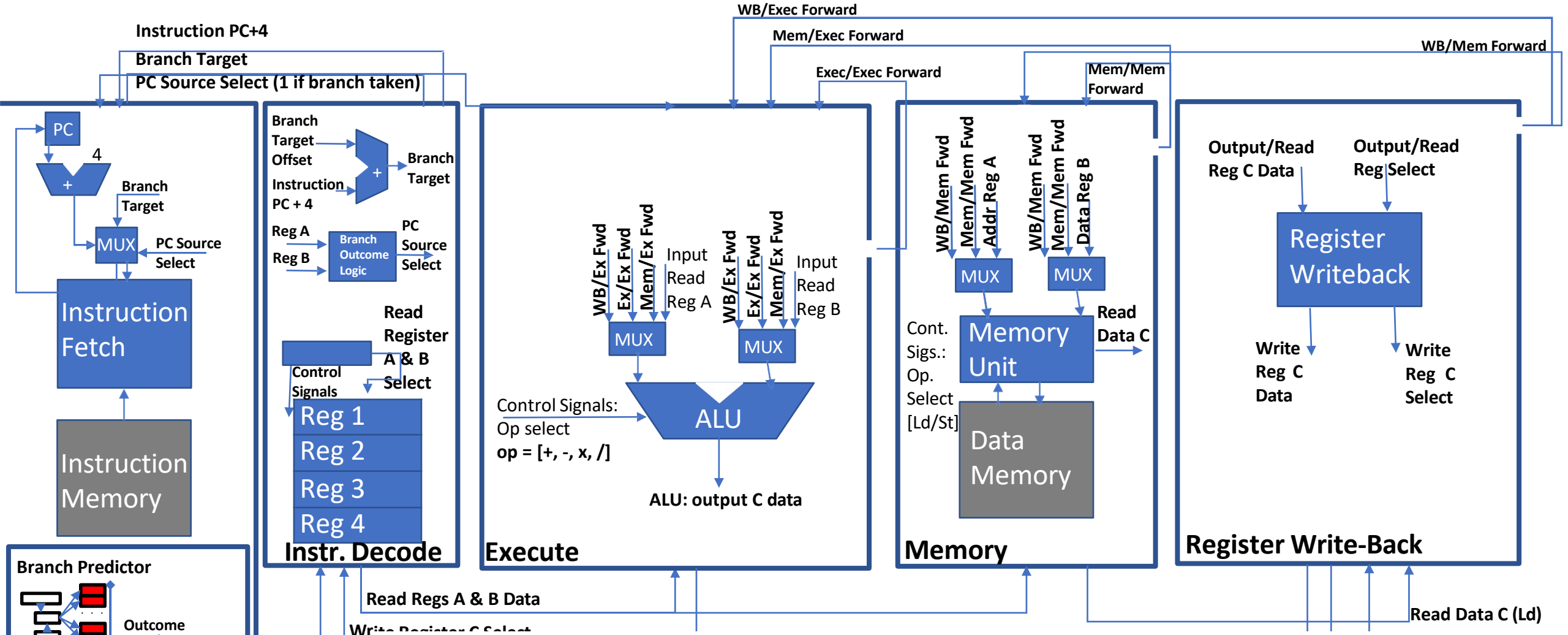
- Performance evaluation basics: metrics, measurement, comparisons
- Benchmarks and benchmarking
- The curse of averages: means, variance, distributions
- Selecting the right metric along which to compare two systems

- Defining a design space
- Pareto Frontiers and Pareto Optimality of a design
- Design space exploration and design iteration

How good is this architecture?

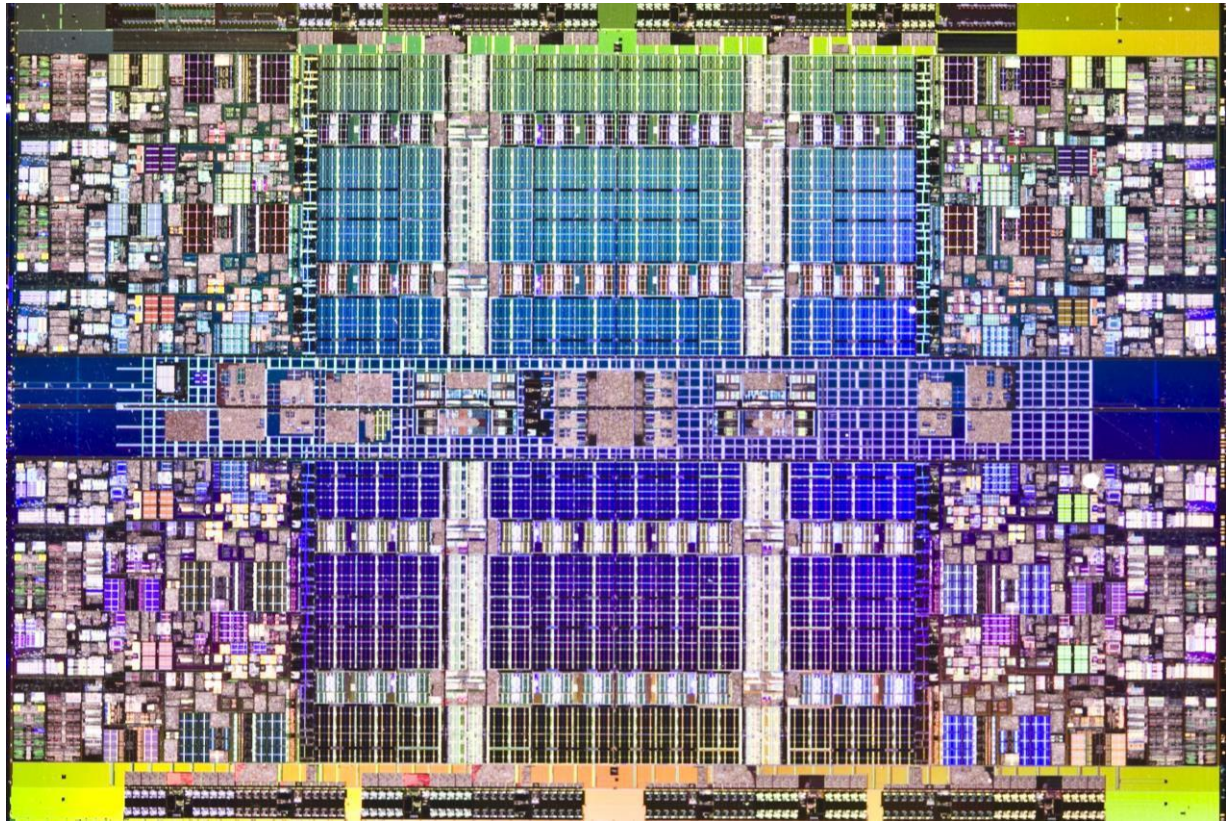


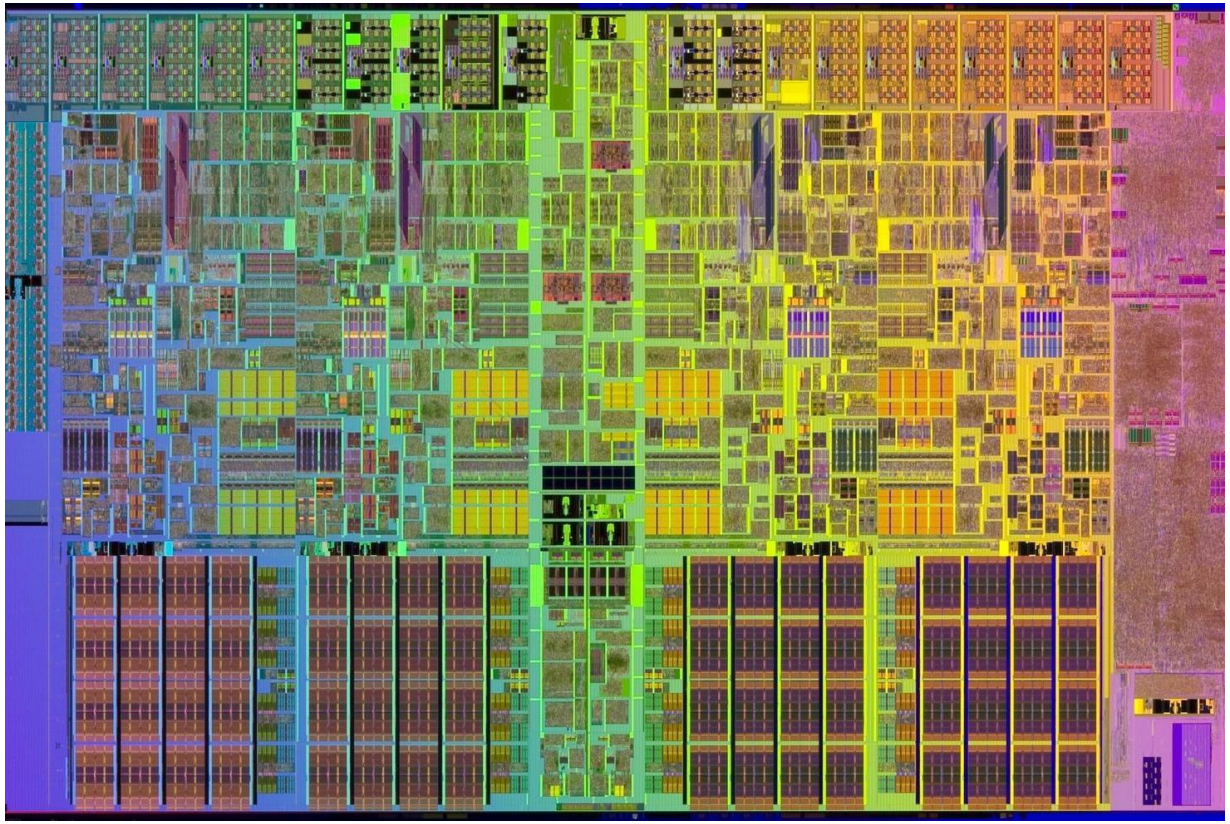
How good is this architecture?



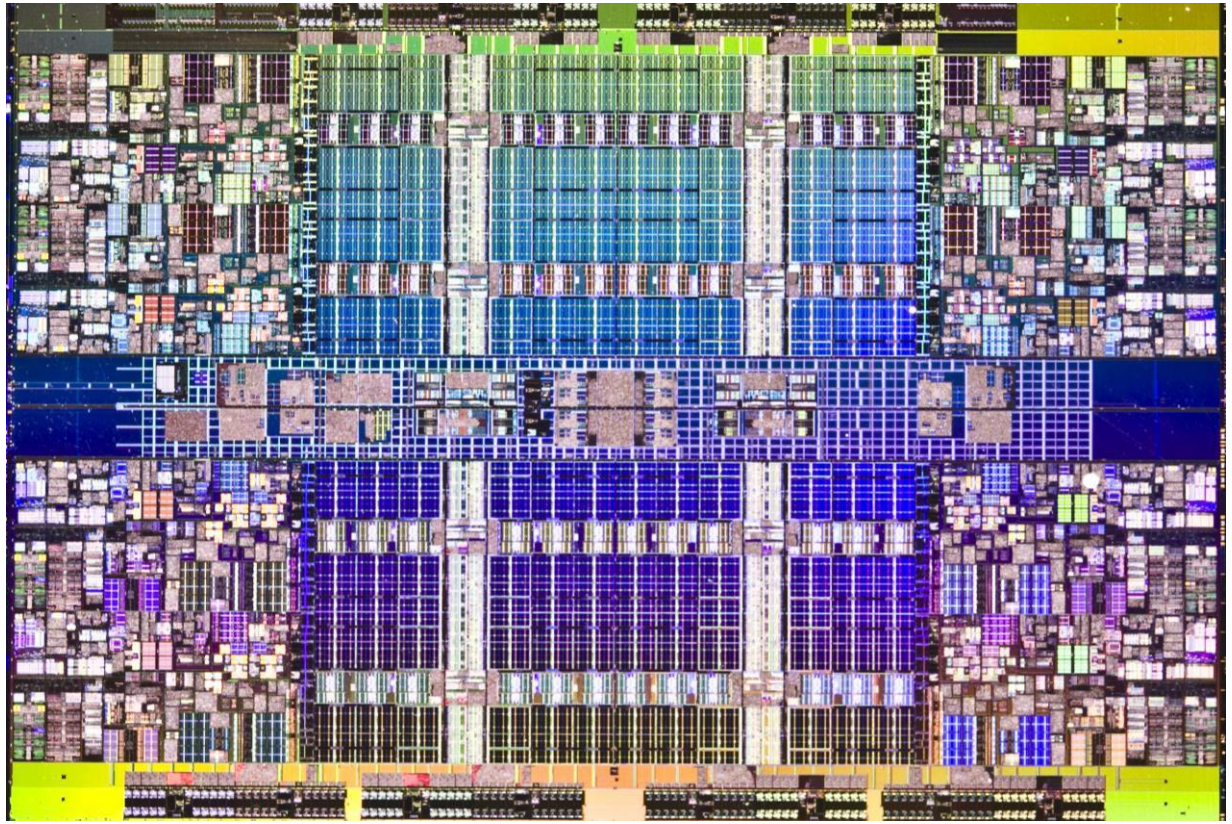
Key Question: Good relative to what?

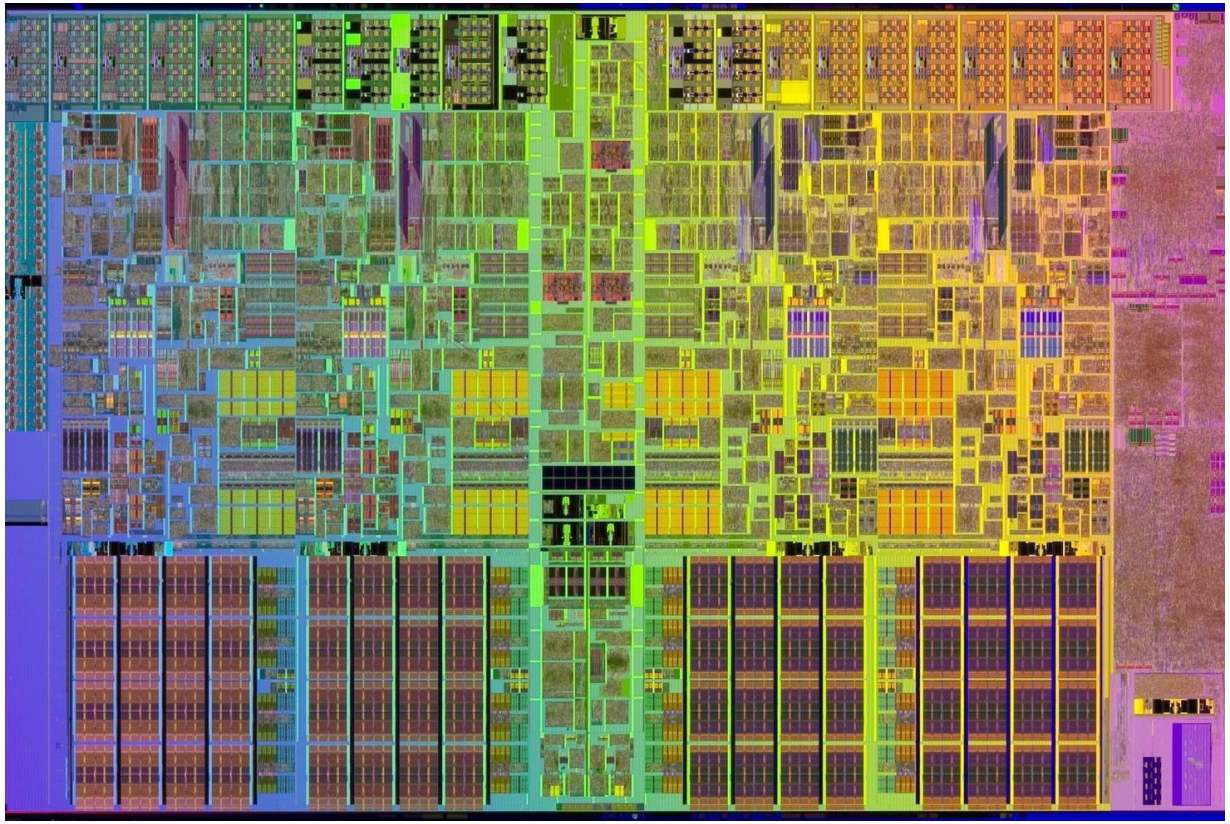
Which architecture is better?





Which architecture is better?





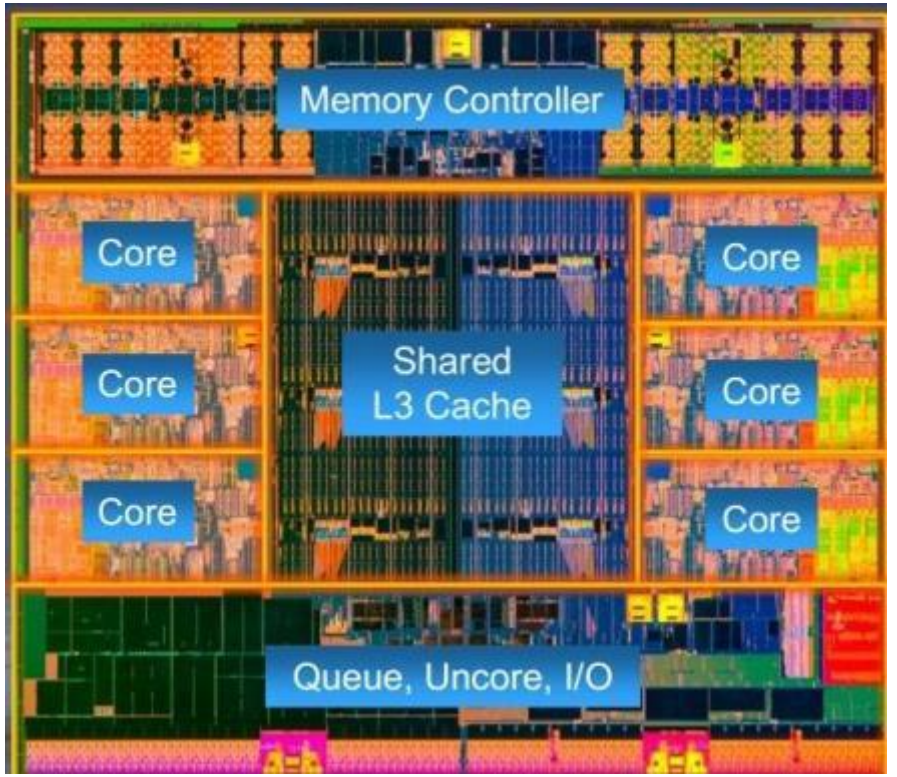
Key Question: What defines “good”?

Intel Core i7-4960X

Area: 257 mm^2

22nm process tech.

3.6GHz



15mm

17.1mm

Intel Pentium 4

Area: 1225 mm^2

180nm process tech.

1.8GHz

35mm

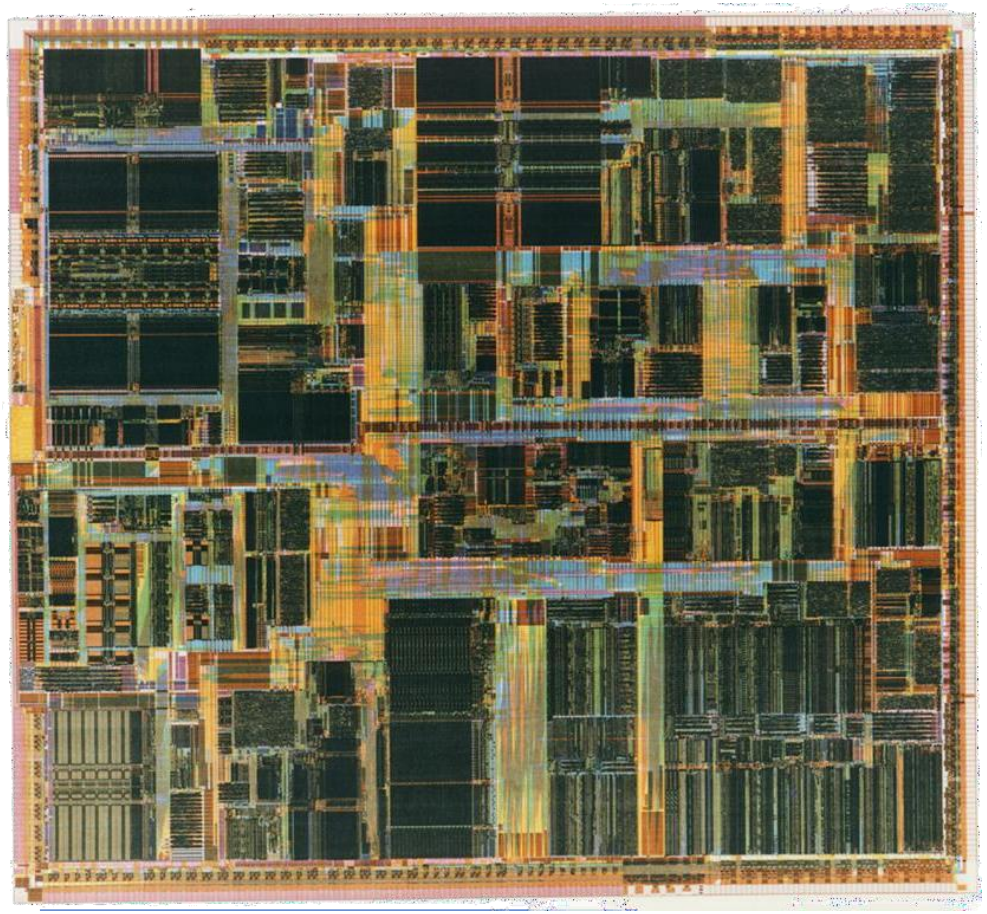
35mm

Intel Pentium Pro / Klamath Falls

Area: 194.8 mm^2

280nm process tech.

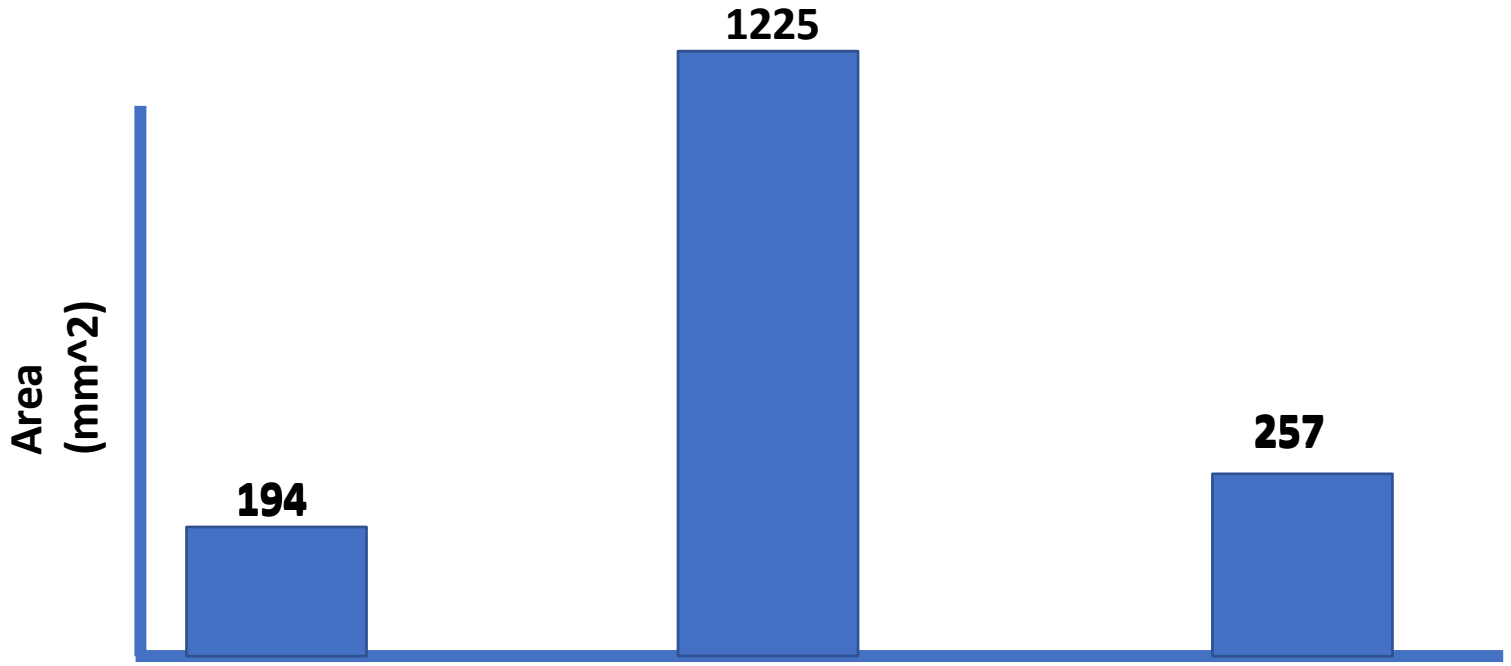
0.3GHz

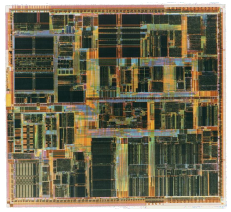


14mm

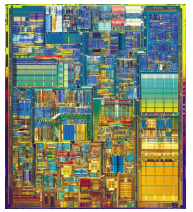
14mm

Comparing Processor Designs

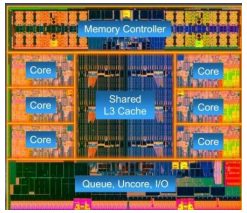




Area: 194.8 mm^2
280nm tech
0.3GHz



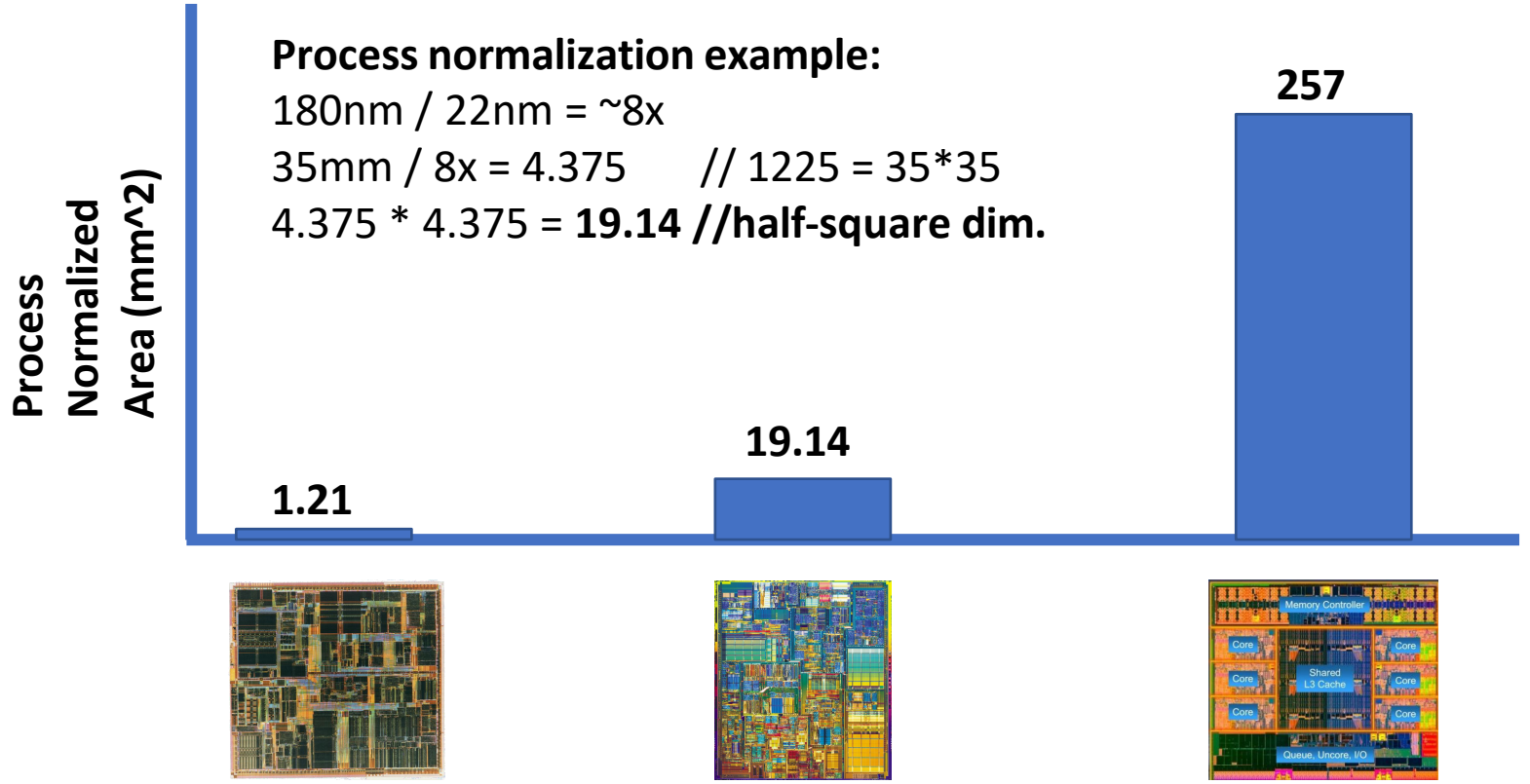
Area: 1225 mm^2
180nm tech
1.8GHz



Area: 257 mm^2
22nm process
tech. 3.6GHz

What do we learn from this comparison?

Comparing Processor Designs



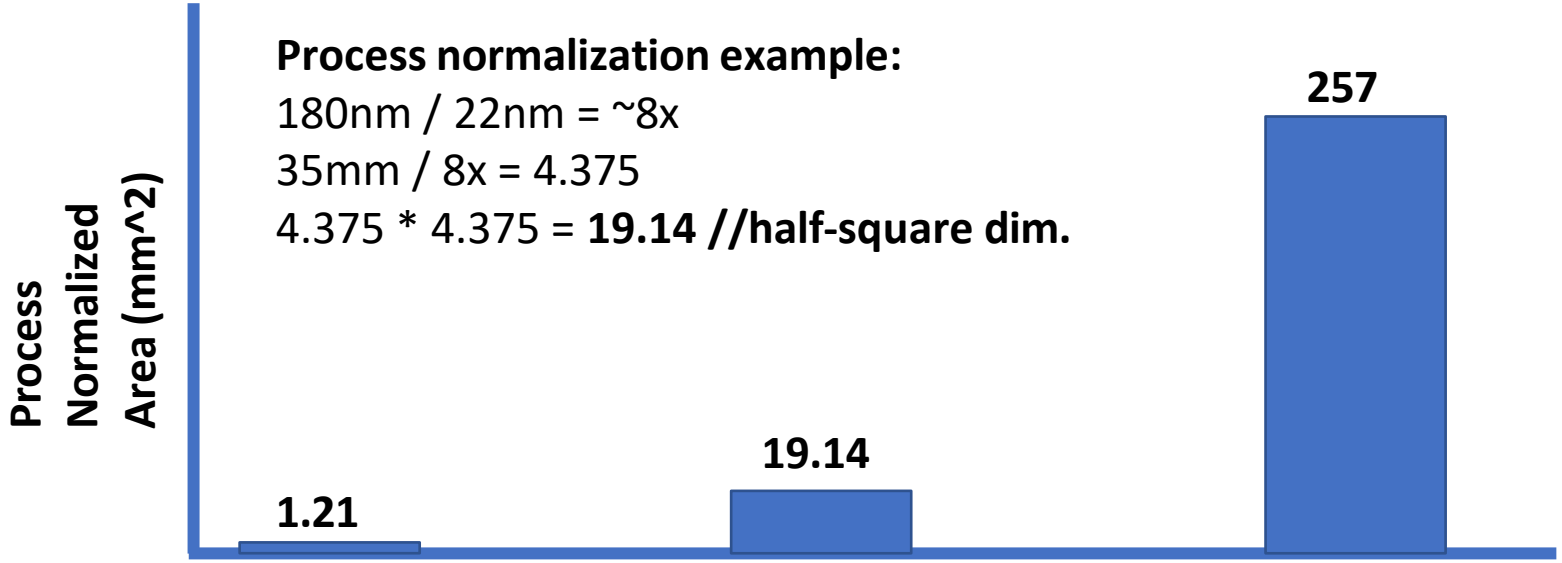
Area: 194.8 mm^2
280nm tech
0.3GHz

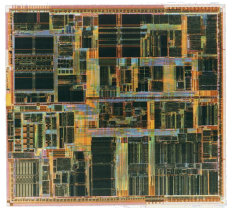
Area: 1225 mm^2
180nm tech
1.8GHz

Area: 257 mm^2
22nm process tech. 3.6GHz

Now what do we learn? What did we change?

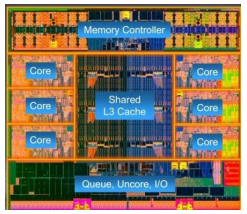
Comparing Processor Designs





Area: 194.8 mm^2
280nm tech
0.3GHz

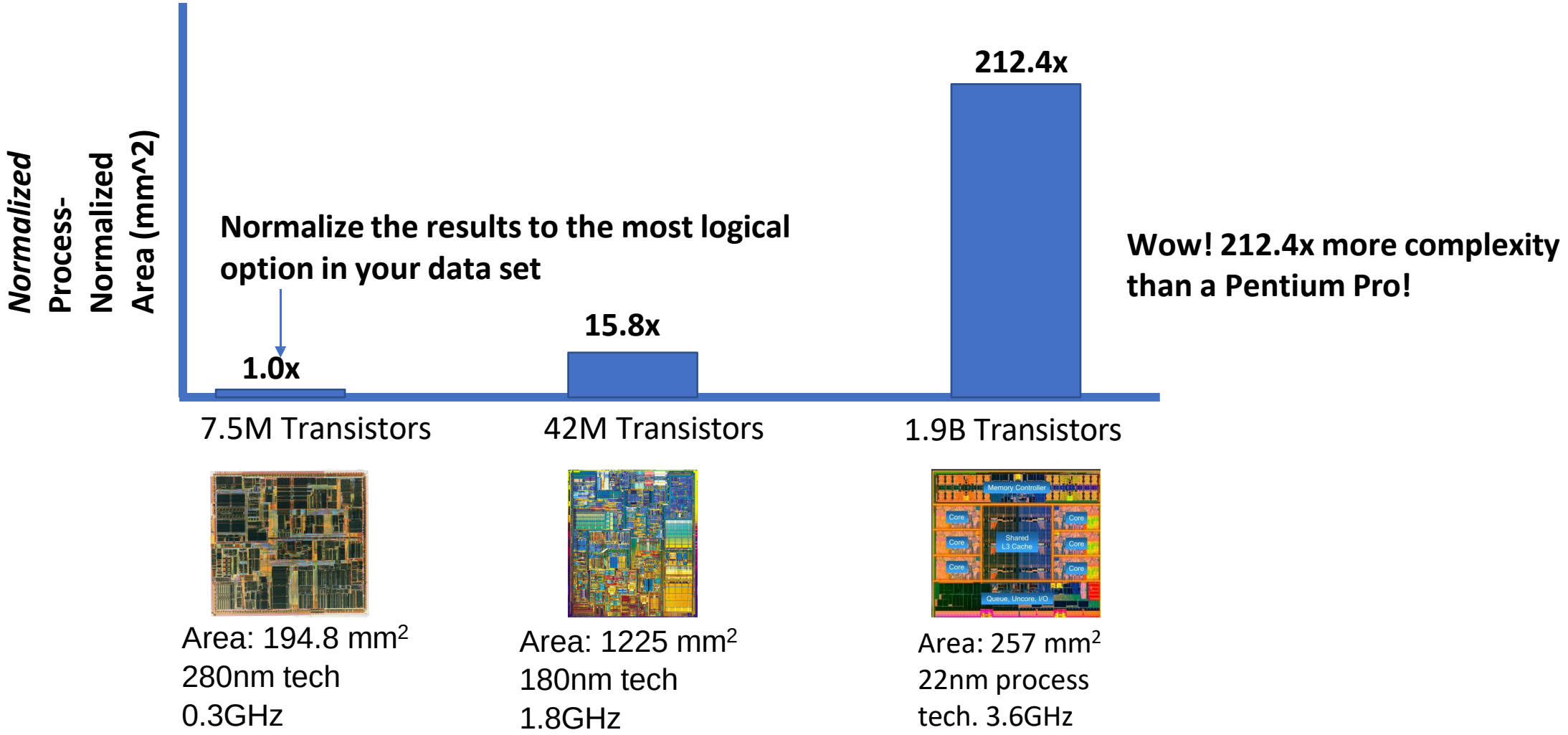
Area: 1225 mm^2
180nm tech
1.8GHz



Area: 257 mm^2
22nm process
tech. 3.6GHz

The **metric** you use should suit the comparison

Comparing Processor Designs



The **relative value** is most important in comparisons

There are lots of metrics to choose from

- Performance:
  - **Run time** or “latency” (seconds to get a result)
  - Throughput (results per second)
  - Instructions per Cycle (IPC) or Cycles per Instruction (CPI)

- Energy:
  - **Total energy**
  - Power (avg. or peak) [we will come back to this one….]
  - Area (absolute or normalized)

- Efficiency:
  - **Performance per power** (GigaOperations Per Second Per Watt: GOPS/W)
  - Operations per Joule (1W = 1J/s, so multiply by J/s, then by 1s)
  - Throughput per unit area (results per second per mm^2)

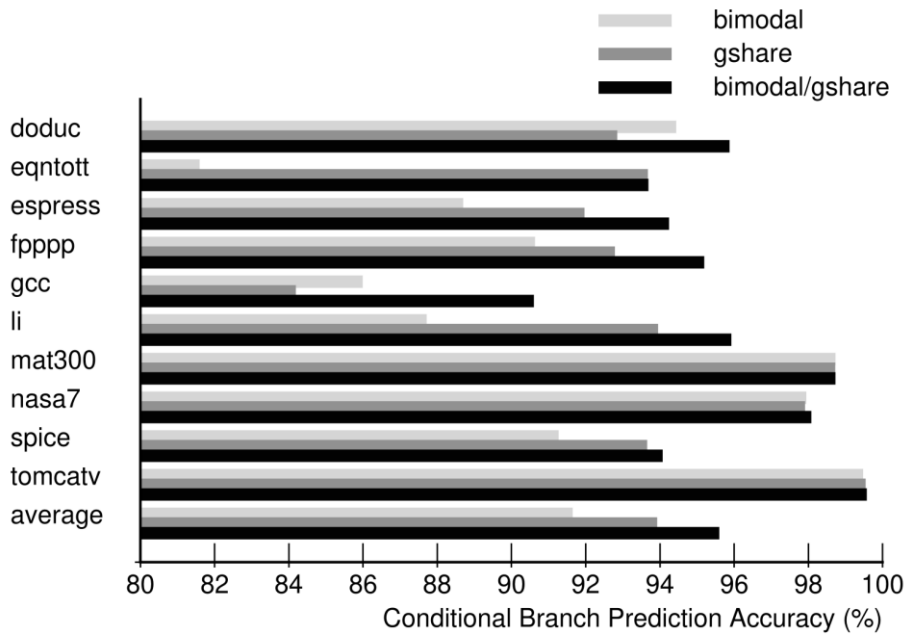
There are lots of metrics to choose from

- **Cache-specific**:
  - Cache Misses per Kilo-Instruction
  - Cache Misses per Kilo-Memory-Instruction
  - Cache Misses per Cache Access
  - **Average Memory Access Time**

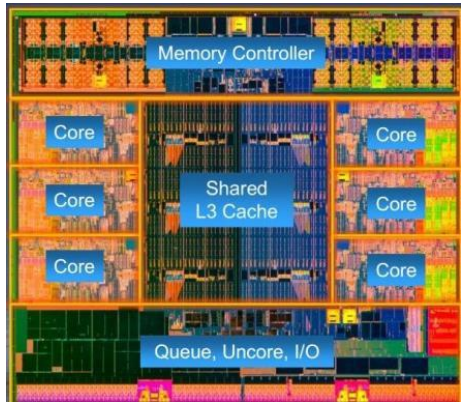
Seriously!?!

- **Control-specific:**
  - Branch prediction accuracy
  - **Branch Mispredictions per Kilo-Instruction**

- **Memory-specific:**
  - **Data throughput (GB/s)**
  - **Memory Latency (seconds per random byte)**

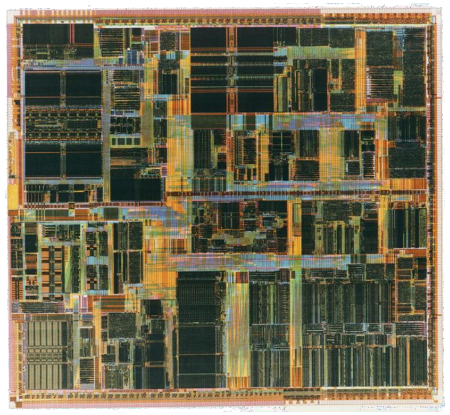


Measuring a System's Performance



Get a system that you want to measure and clean it up and isolate it from everything else in the world

Find a workload that you can run on your system and that will produce a meaningful result



Choose an appropriate baseline system to which to compare your results to understand their relative merit

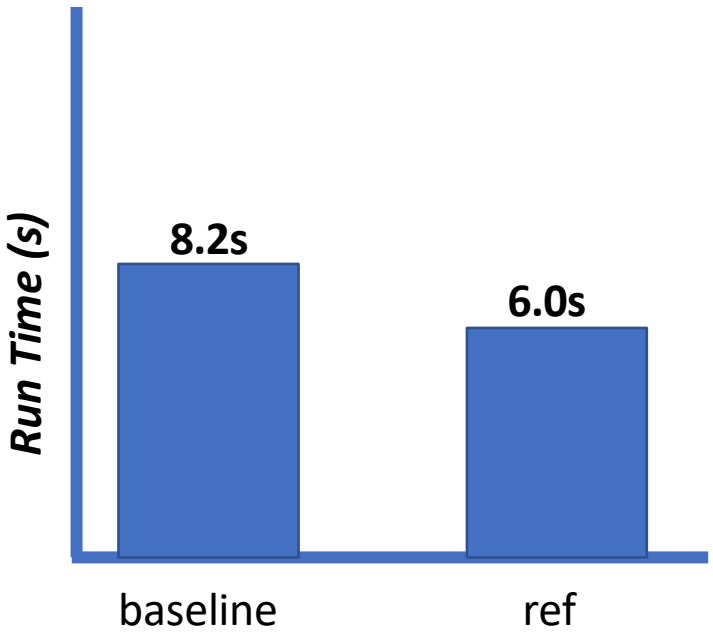
Measure your baseline…

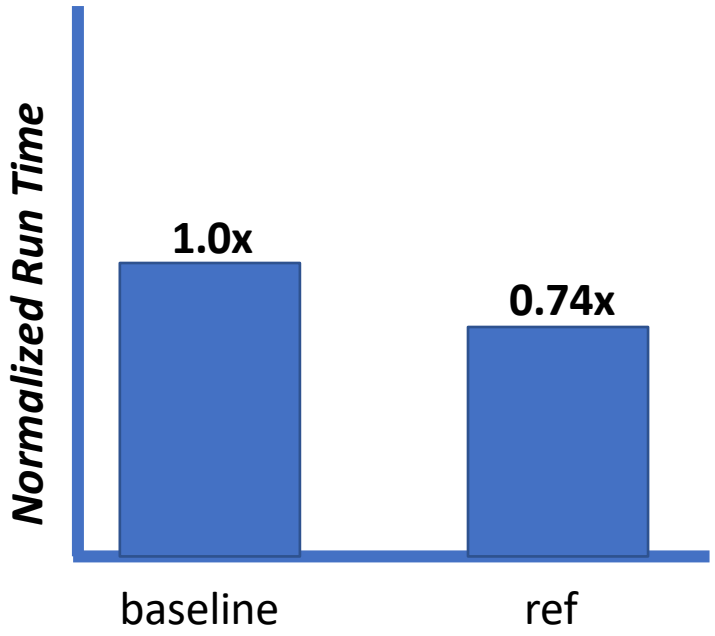
```
blucia@Nuugaatsiaq:~/cvsandbox/18-344/graph-optimization$ time ./handout 100m.
g 100m.handout.new.csr
Loading 100000000 edges
Loading [100m.g]...Done.
Counting neighbors...Accumulating neighbor counts...Done.
Populating neighbors...Done.
Ejecting CSR...Writing out CSR data to [100m.handout.new.csr]...Done.
Done.

real    0m27.546s
user    0m8.207s
sys     0m1.069s
```

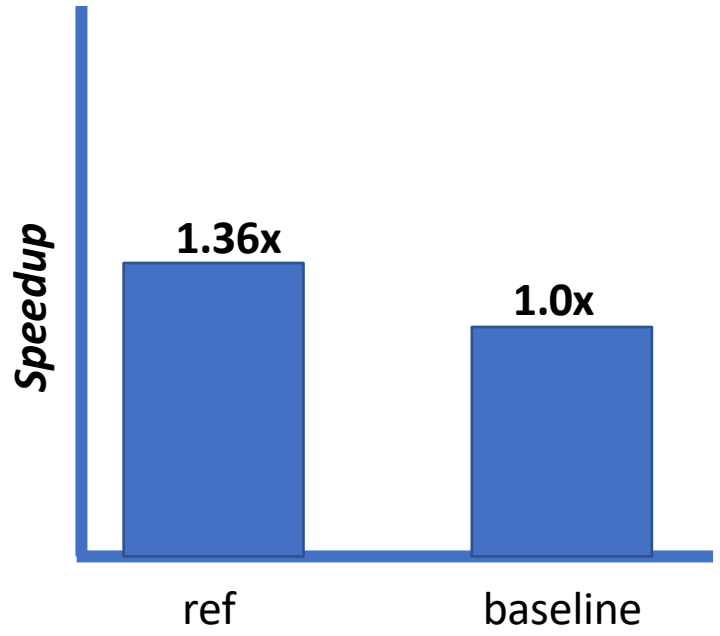
…measure your system…

```
blucia@Nuugaatsiaq:~/cvsandbox/18-344/graph-optimization$ time ./ref 100m.g 10
0m.ref.new.csr
Loading 100000000 edges
Loading [100m.g]...Done.
Counting neighbors...Accumulating neighbor counts...Done.
Populating neighbors...Done.
Ejecting CSR...Writing out CSR data to [100m.ref.new.csr]...Done.
Done.

real    0m23.878s
user    0m6.078s
sys     0m2.039s
```

…and compare the results

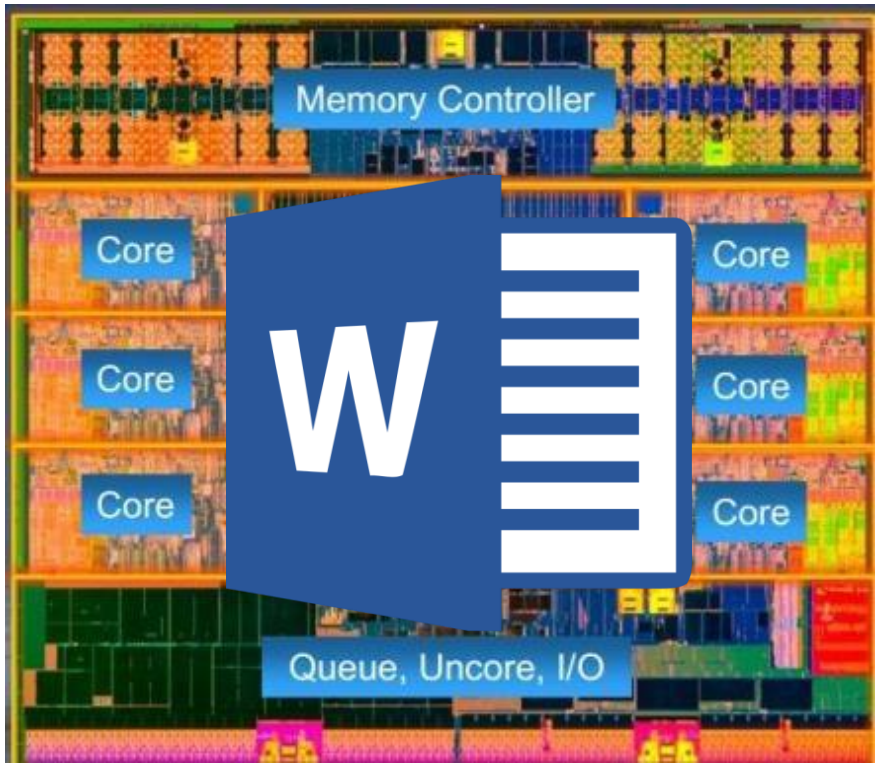






These are equivalent results, with different presentations

Choosing your measurement workload
The “Benchmark Problem”



What are we looking for in a benchmark program?

- **Representative** of typical workloads for system
- **Runs long enough** to measure precisely
- Runs in **reasonable time** for repeated trials
- Runs with **deterministic** metric values
- Runs w/ **no excessively large inputs** or I/O
- Runs with **no human intervention**

Benchmark Suites

The Standard Performance Evaluation Corporation

- Founded 1988 by workstation computer manufacturers
- Goal: eliminate hype misinformation, replace hype with hard data, eliminate **“apples-to-oranges” comparison** of computer systems
- Methodology: provide standardized suite of source code that a benchmarker (a human) can compile for their system, tune for best results, and run for an official benchmark performance score comparable to others.
- Updates benchmark suite periodically to keep it in line with the market, i.e. the workloads driving the sales.

What are these benchmark programs?

SPECspeed®2017 Integer	Language [1]	KLOC [2]	Application Area
600.perlbench_s	C	362	Perl interpreter
602.gcc_s	C	1,304	GNU C compiler
605.mcf_s	C	3	Route planning
620.omnetpp_s	C++	134	Discrete Event simulation - computer network
623.xalancbmk_s	C++	520	XML to HTML conversion via XSLT
625.x264_s	C	96	Video compression
631.deepsjeng_s	C++	10	Artificial Intelligence: alpha-beta tree search (Chess)
641.leela_s	C++	21	Artificial Intelligence: Monte Carlo tree search (Go)
648.exchange2_s	Fortran	1	Artificial Intelligence: recursive solution generator (Sudoku)
657.xz_s	C	33	General data compression

MLPerf ("MLCommons") Benchmarks

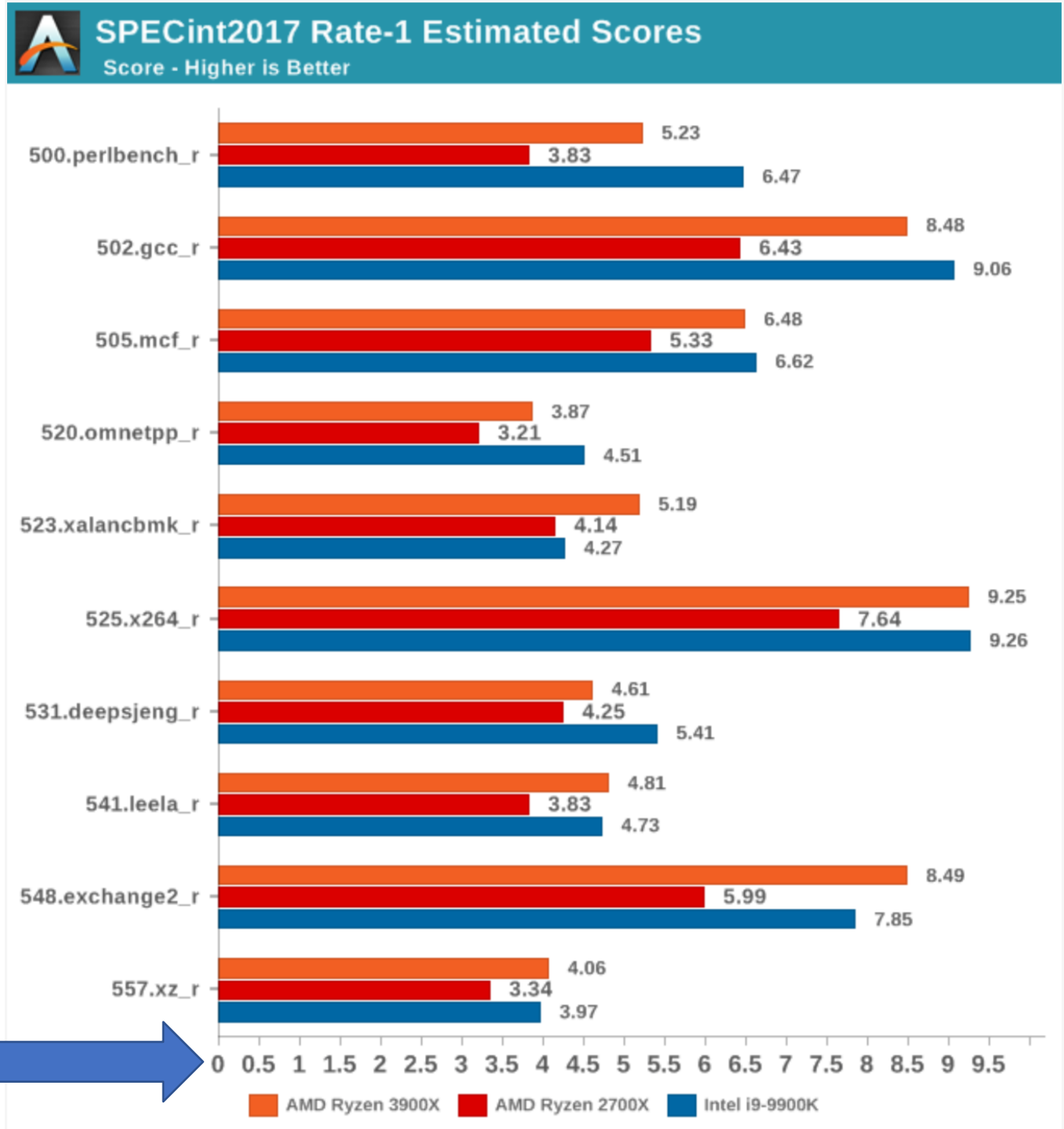
TABLE I
ML TASKS IN MLPERF INFERENCE V0.5. EACH ONE REFLECTS CRITICAL COMMERCIAL AND RESEARCH USE CASES FOR A LARGE CLASS OF SUBMITTERS, AND TOGETHER THEY COVER A BROAD SET OF COMPUTING MOTIFS (E.G., CNNS AND RNNS).

AREA	TASK	REFERENCE MODEL	DATA SET	QUALITY TARGET
VISION	IMAGE CLASSIFICATION (HEAVY)	RESNET-50 V1.5 25.6M PARAMETERS 8.2 GOPS / INPUT	IMAGENET (224X224)	99% OF FP32 (76.456%) TOP-1 ACCURACY
VISION	IMAGE CLASSIFICATION (LIGHT)	MOBILENET-V1 224 4.2M PARAMETERS 1.138 GOPS / INPUT	IMAGENET (224X224)	98% OF FP32 (71.676%) TOP-1 ACCURACY
VISION	OBJECT DETECTION (HEAVY)	SSD-RESNET-34 36.3M PARAMETERS 433 GOPS / INPUT	COCO (1,200X1,200)	99% OF FP32 (0.20 MAP)
VISION	OBJECT DETECTION (LIGHT)	SSD-MOBILENET-V1 6.91M PARAMETERS 2.47 GOPS / INPUT	COCO (300X300)	99% OF FP32 (0.22 MAP)
LANGUAGE	MACHINE TRANSLATION	GNMT 210M PARAMETERS	WMT16 EN-DE	99% OF FP32 (23.9 SACREBLEU)

What are these benchmark programs?

SPECspeed®2017 Integer	Language [1]	KLOC [2]	Application Area
600.perlbench_s	C	362	Perl interpreter
602.gcc_s	C	1,304	GNU C compiler
605.mcf_s	C	3	Route planning
620.omnetpp_s	C++	134	Discrete Event simulation - computer network
623.xalancbmk_s	C++	520	XML to HTML conversion via XSLT
625.x264_s	C	96	Video compression
631.deepsjeng_s	C++	10	Artificial Intelligence: alpha-beta tree search (Chess)
641.leela_s	C++	21	Artificial Intelligence: Monte Carlo tree search (Go)
648.exchange2_s	Fortran	1	Artificial Intelligence: recursive solution generator (Sudoku)
657.xz_s	C	33	General data compression

Allows us to compare different machines?



What is this axis showing?

SPEC Baselines to a Reference Machine

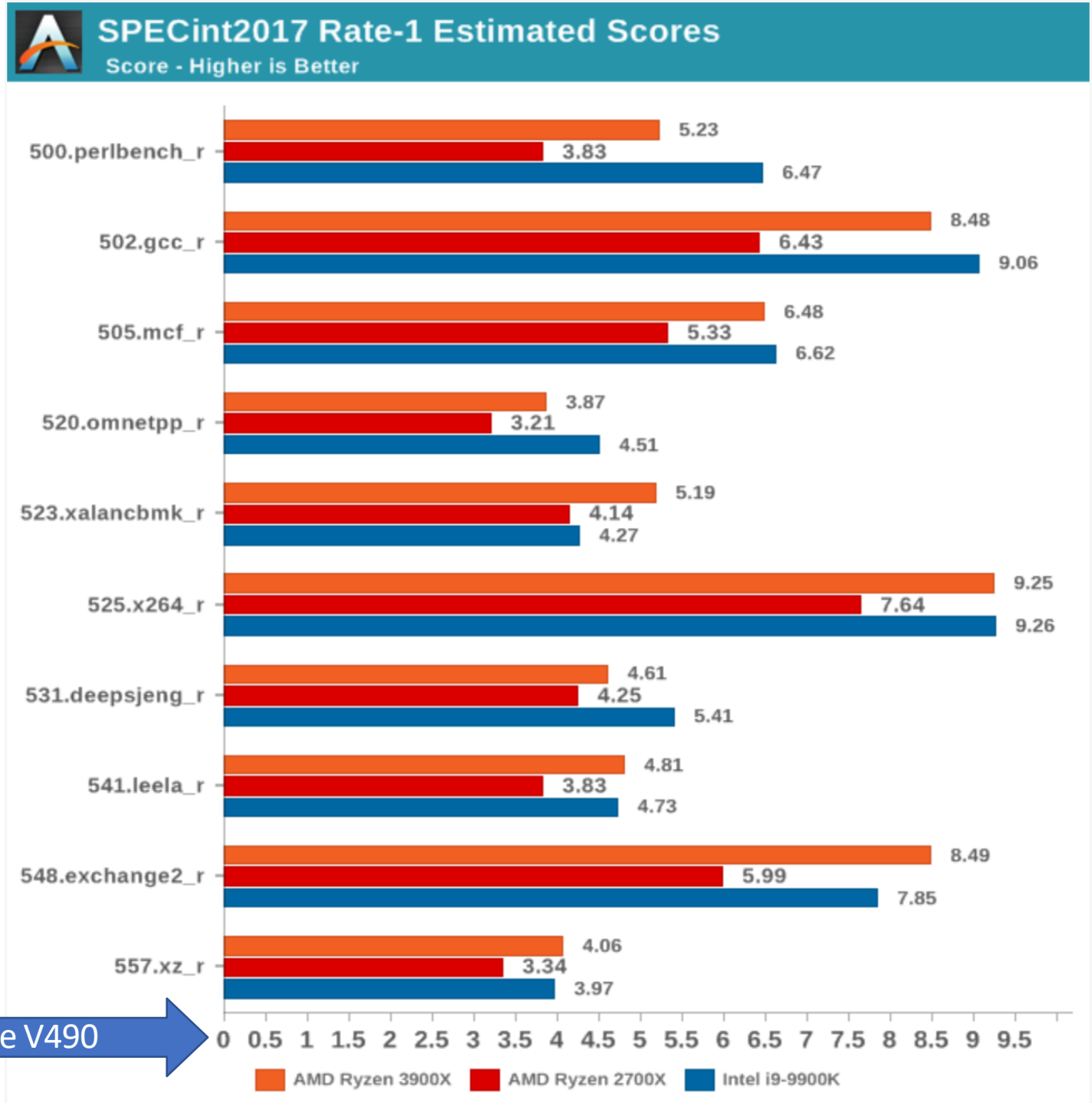
Oracle Sun Fire V490
Competitive 2000s-era server
4-Socket UltraSPARC IV architecture
64GB Max Memory Capacity
Last shipped April 2009

For SPEC2017, this machine represents 1.0 SPECMarks

Why compare to some random machine from 2001?

Allows us to compare arbitrary machines!

SPECMarks: Normalized throughput relative to Sun Fire V490



The SPEC Reference Machine Challenge

- A current competitor won't fly
  - It is an industry benchmark.
  - Do you want your machine to be the slow 1.0?
  - Do you want everyone talking about a competitor's product as if it is the current gold standard

- An unfamiliar machine is an unintuitive metric

- An imaginary machine is worse

- An older machine may be poorly suited for current workloads
  - It isn't just ops/sec.
  - Consider a modern large working set workload run on an older machine with a smaller cache. How does one compare?

- A recent "venerable workhorse" may be the best option

A Few Other Presently Relevant Benchmarks

- Cloudsuite
  - Data center works (may be a little less relevant than a few years ago)
- MLPerf
  - Cloud-based training and inference performance
- GAP Code (Berkeley)
  - Data center works (may be a little less relevant than a few years ago)

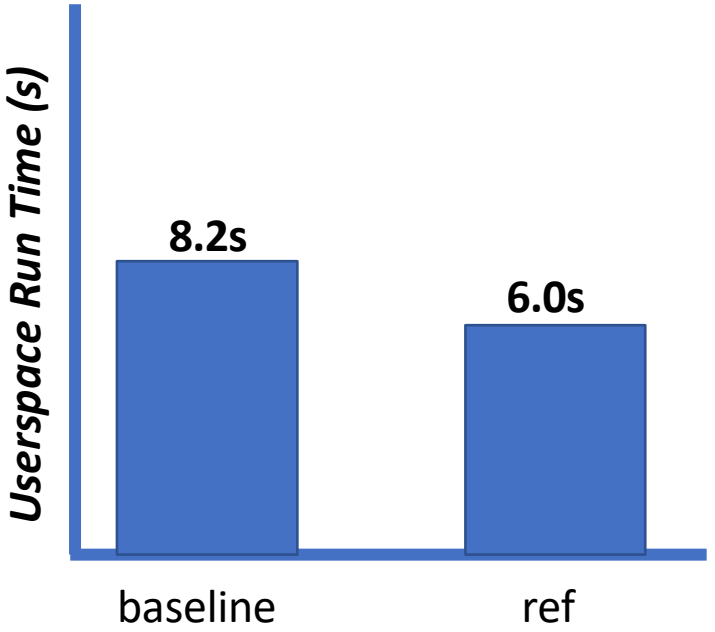
It is sometimes appropriate to propose a new benchmark tuned to emerging concerns, but results are typically also reported for well-accepted benchmark(s) until the research community understands the proposed benchmark, observes a critical mass of research data based upon it, and accepts its relevance.

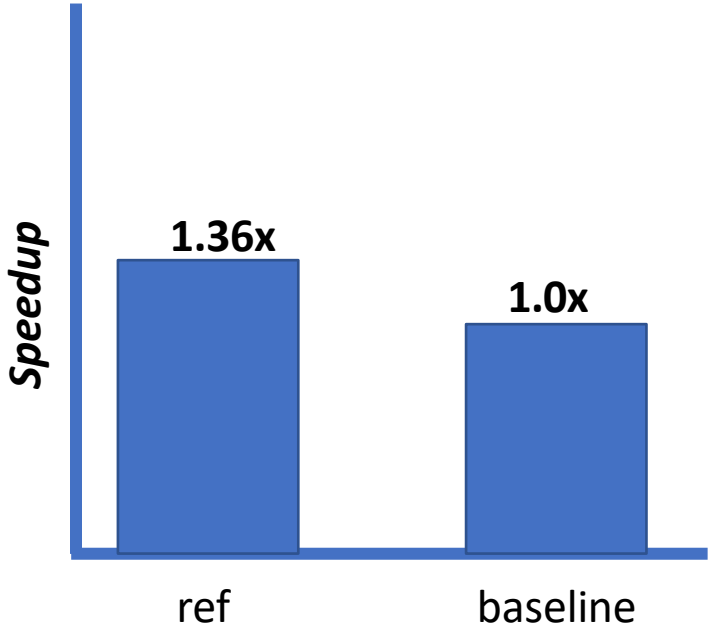
Lies, Damned Lies, and Statistics (Twain/Diraeli)

- Comparing to the wrong thing can make anything look good
  - Comparing an improvement of a feature to the absence of a feature, e.g. a new branch predictor to none at all vs comparing it to a well-accepted best-of-class
- Comparing in a nonsensical environment can make anything look good
  - Workload too large for cache on reference machine
  - Comparing single core to multicore and letting bus traffic interfere
- Choosing a benchmark not relevant to the audience or problem
  - E.g., just a SPEC benchmark, when the use case is a novel workload for which the chosen benchmark, though well accepted for other purposes, is not relevant.
- Managing the data
  - Cherry-picking runs (when there is environmental interference)
  - Not excluding outliers (when there is environmental interference)
  - Averaging or otherwise summarizing inappropriately

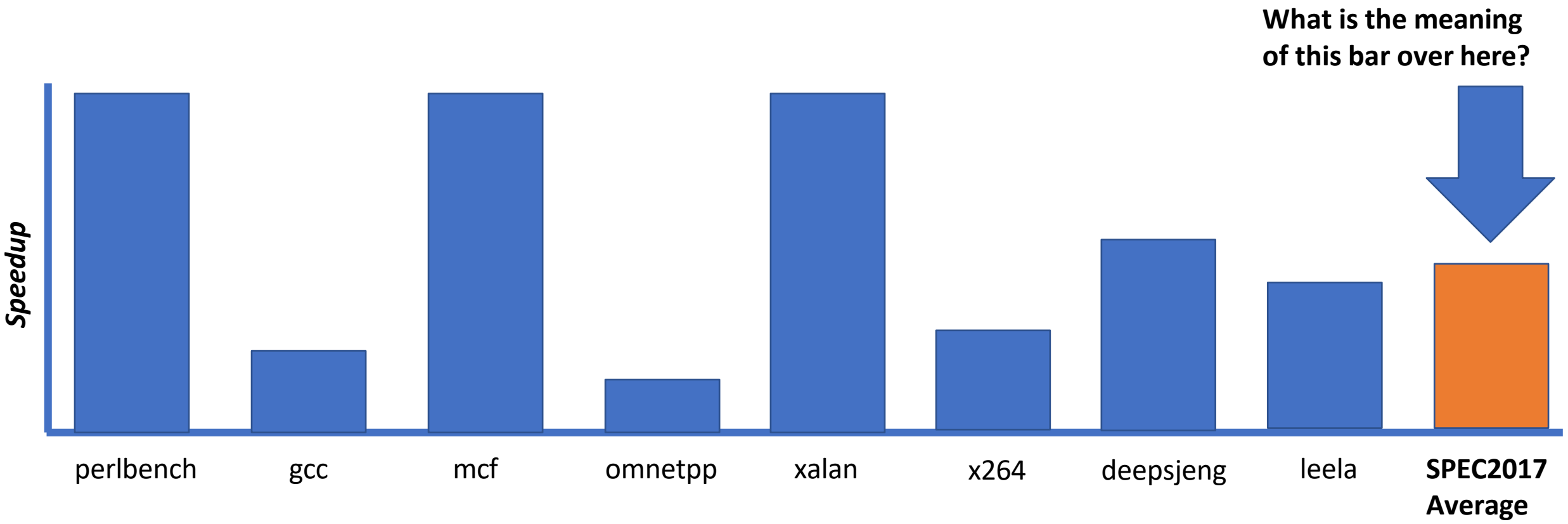
Practical Normalized Performance Measurement

- Choose appropriate baseline system
  - Not going to use Oracle Sun Fire V490 unless you're submitting to SPEC
- Select a highly optimized baseline
  - Avoid "Straw Man" comparisons
- Select a comparable baseline
  - Avoid "apples-to-oranges" comparisons





Single Number Performance Summaries



On Computing Averages: Options

$$ArithmeticMean\,(a_1, a_2, a_3, \ldots, a_N) = \frac{\sum_i^N a_i}{N}$$

$$GeometricMean\,(a_1, a_2, a_3, \ldots, a_N) = \sqrt[N]{\prod_i^N a_i}$$

$$HarmonicMean\,(a_1, a_2, a_3, \ldots, a_N) = \frac{N}{\sum_i^N \frac{1}{a_i}}$$

Amean – Good for averaging times

$$ArithmeticMean\,(a_1, a_2, a_3, \ldots, a_N) = \frac{\sum_{i}^{N} a_i}{N}$$

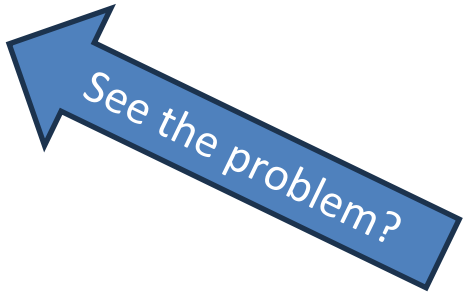
- Averages all data points equally. Sums them up and divides by number of fata points
- Good for times.

Geometric Mean – good for speedups

$$GeometricMean\,(a_1, a_2, a_3, \ldots, a_N) = \sqrt[N]{\prod_i^N a_i}$$

Program 1: 0.5x speedup
Program 2: 2.0x speedup
Average speedup: 1.0 (one is half as fast, one is twice as fast)

Amean(0.5,2.0) = (0.5+2.0) / 2 = 2.5 / 2 = 1.25x



Gmean(0.5,2.0) = sqrt(0.5 * 2.0) = Sqrt(1.0) = 1.0

Harmonic Mean – Good for rates

$$HarmonicMean\,(a_1, a_2, a_3, \ldots, a_N) = \frac{N}{\sum_i \frac{1}{a_i}}$$

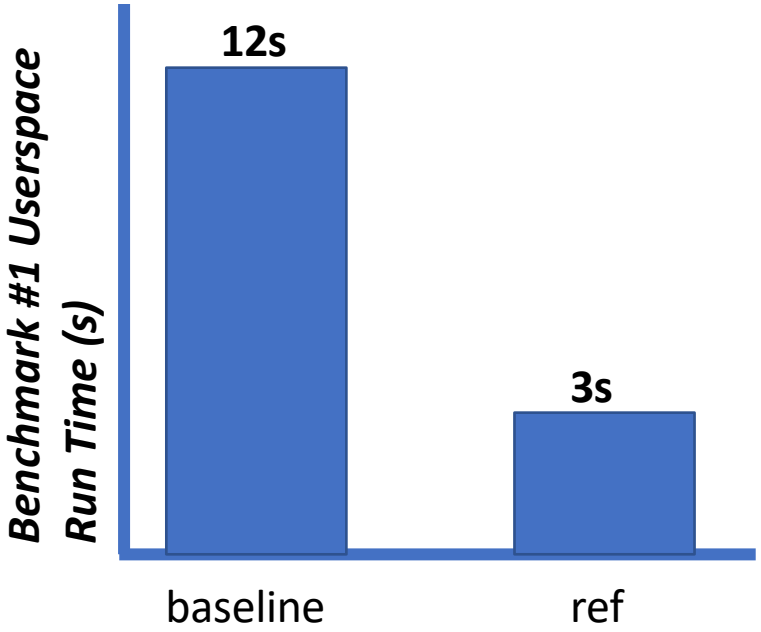
F() has N instructions

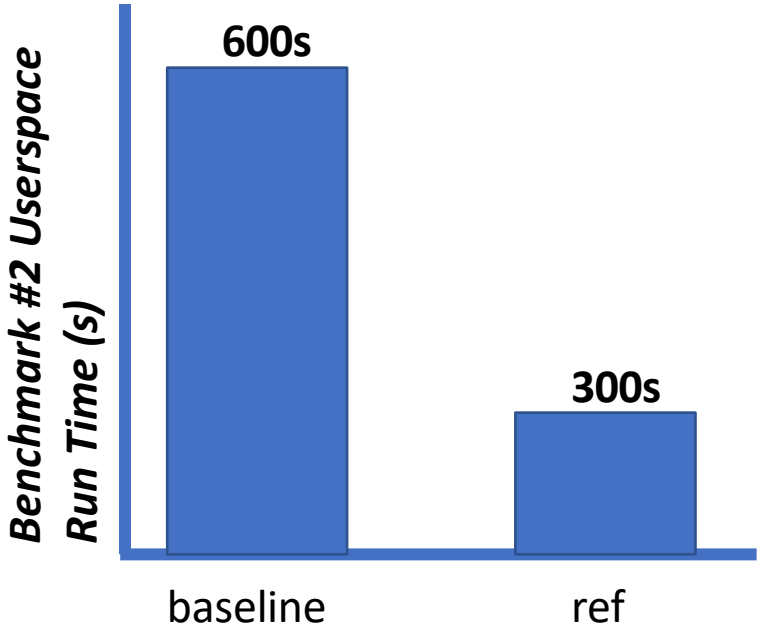
N instructions at 60GOPS
N same instructions at 20GOPS
Total time = N/60 + N/20 = N/15; N=60 -> 4s.

Amean: (60GOPS + 20 GOPS) / 2 = 40GOPS
Total time using Amean: 2N / 40; N = 60 -> 120/40 = 3s (incorrect)

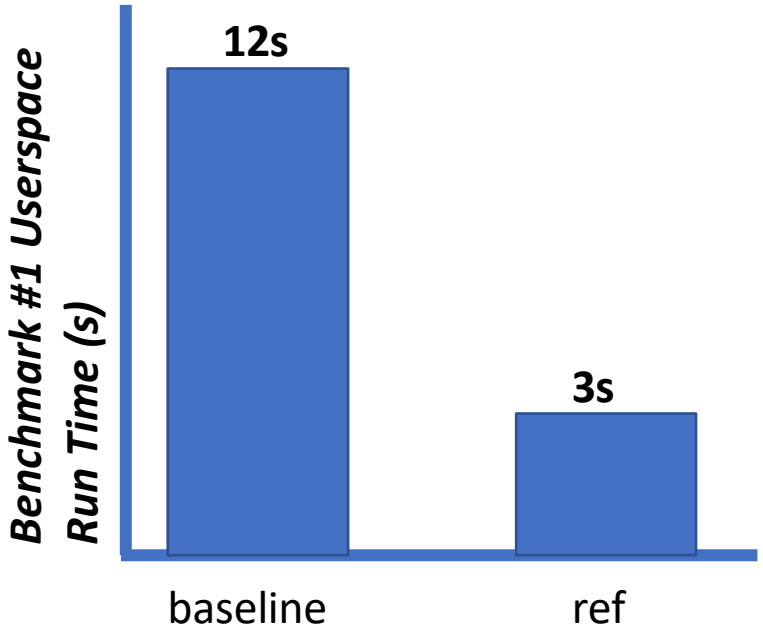
Hmean: Avg GOPS = Total Instructions / Sum of per-call times =
2N / (N/60 + N/20) = 2 / (1/60 + 1/20) = 2 / (1 / 15) = 30 GOPS
Total time using Hmean: 2N / 30; N = 60 -> 120/30 = 4s (correct)

The problem with single-number performance summaries by example

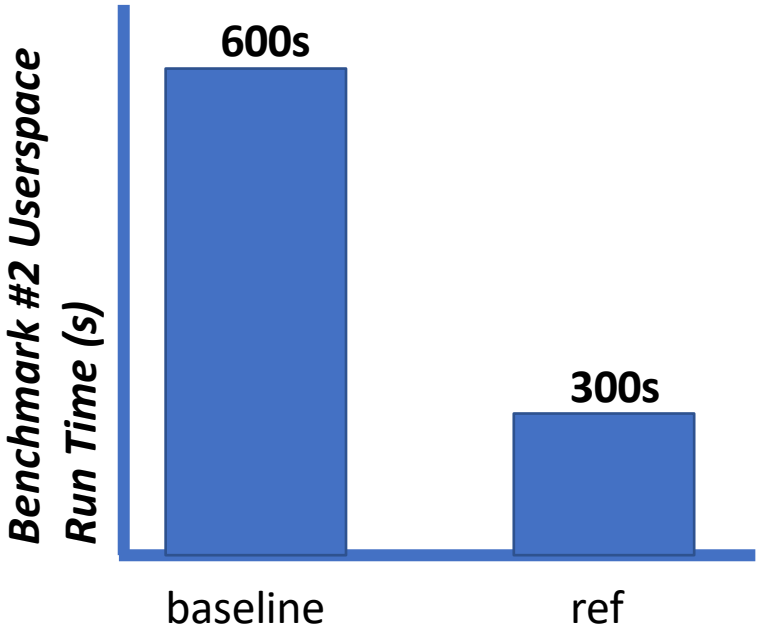




Thinking in ratios



Speedup = 12s / 3s = 4x



Speedup = 600s / 300s = 2x

Unweighted Average Speedup (gmean of speedups) = sqrt(4x * 2x) = 2.82x (No benchmark is more important or prevalent than any other in the wild despite differences in time during tests – unweighted average.)

Time-weighted Average Speedup (amean of times) = 612s / 303s = 2.02x (Runtime of each test indicates its prevalence/importance in the wild. Tests that run longer count for more – weighted average. This only makes sense if workloads balance out in this proportion in the wild. Otherwise can weight by prevalence, etc.)

Isolating your measurement environment

Isolating your measurement environment

```
top - 16:16:03 up  2:08,  0 users,  load average: 0.00, 0.00, 0.00
Tasks:   5 total,   1 running,   4 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.1 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
MiB Mem :  12718.9 total,  12372.5 free,     83.1 used,    263.3 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.  12394.2 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0     892    584    516 S   0.0   0.0   0:00.02 init
    6 root      20   0     892     88     20 S   0.0   0.0   0:00.00 init
    7 root      20   0     892     88     20 S   0.0   0.0   0:01.20 init
    8 blucia    20   0   10280   5408   3488 S   0.0   0.0   0:00.71 bash
  508 blucia    20   0   10868   3724   3216 R   0.0   0.0   0:00.00 top
```

First check the basics (like no one is gamin’ on the rig)

Unintuitive Sources of Performance Interference

Producing Wrong Data Without Doing Anything Obviously Wrong!

Todd Mytkowicz Amer Diwan
Department of Computer Science
University of Colorado
Boulder, CO, USA
{mytkowit,diwan}@colorado.edu

Matthias Hauswirth
Faculty of Informatics
University of Lugano
Lugano, CH
Matthias.Hauswirth@unisi.ch

Peter F. Sweeney
IBM Research
Hawthorne, NY, USA
pfs@us.ibm.com

```
blucia@Nuugaatsiaq:~$ env
SHELL=/bin/bash
PINPATH=/home/blucia/cvsandbox/pin/pin-3.18-98332-gaebd7b1e6-gcc-linux
WSL_DISTRO_NAME=Ubuntu-20.04
NAME=Nuugaatsiaq
PWD=/home/blucia
LOGNAME=blucia
MOTD_SHOWN=update-motd
HOME=/home/blucia
LANG=C.UTF-8
WSL_INTEROP=/run/WSL/632_interop
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=3
7;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;3
1:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz
=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio
=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bm
p=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.p
cx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.
nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv
=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc
=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
PIN_ROOT=/home/blucia/cvsandbox/pin/pin-3.18-98332-gaebd7b1e6-gcc-linux
PIN_HOME=/home/blucia/cvsandbox/pin/pin-3.18-98332-gaebd7b1e6-gcc-linux
LESSCLOSE=/usr/bin/lesspipe %s %s
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=blucia
DISPLAY=172.17.80.1:0
SHLVL=1
WSLENV=
XDG_DATA_DIRS=/usr/local/share:/usr/share:/var/lib/snapd/desktop
PROJECT_ROOT=/home/blucia/cvsandbox/panic
PATH=/home/blucia/.local/bin:/home/blucia/cvsandbox/pin/pin-3.18-98332-gaebd7b1e6-gcc-
linux:/home/blucia/.local/bin/:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/mnt/c/Program
Files/WindowsApps/CanonicalGroupLimited.Ubuntu20.04onWindows_2004.2021.610.0_x64__79rhkp1fndgsc:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System3
2/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/mnt/c/WINDOWS/System32/OpenSSH/:/mnt/c/Program Files (x86)/PDFtk Server/bin/:/mnt/c/Program
Files/MIT/Kerberos/bin:/mnt/c/Program Files/OpenAFS/Common:/mnt/c/Program
Files/OpenAFS/Client/Program:/mnt/c/Users/bluci/AppData/Local/Microsoft/WindowsApps:/snap/bin
HOSTTYPE=x86_64
RISCV=/home/blucia/cvsandbox/panic-riscv-tools/riscv-tools
_=/usr/bin/env
blucia@Nuugaatsiaq:~$
```

```
blucia@Nuugaatsiaq:~$ env
SHELL=/bin/bash
PINPATH=/home/blucia/cvsandbox/pin/pin-3.18-98332-gaebd7b1e6-gcc-linux
WSL_DISTRO_NAME=Ubuntu-20.04
NAME=Nuugaatsiaq
PWD=/home/blucia
LOGNAME=blucia
MOTD_SHOWN=update-motd
HOME=/home/blucia
LANG=C.UTF-8
WSL_INTEROP=/run/WSL/632_interop
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=3
7;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;3
1:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz
=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio
=01;31:*.7z=01;31:*.r                                                                                                                    gif=01;35:*.bm
p=01;35:*.pbm=01;35:                                                                                                                     .mng=01;35:*.p
cx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.
nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv
=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc
=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
PIN_ROOT=/home/blucia/cvsandbox/pin/pin-3.18-98332-gaebd7b1e6-gcc-linux
PIN_HOME=/home/blucia/cvsandbox/pin/pin-3.18-98332-gaebd7b1e6-gcc-linux
LESSCLOSE=/usr/bin/lesspipe %s %s
TERM=xterm-256color
LESSOPEN=| /usr/bin/l
USER=blucia
DISPLAY=172.17.80.1:0
SHLVL=1
WSLENV=
XDG_DATA_DIRS=/usr/local/share:/usr/share:/var/lib/snapd/desktop
PROJECT_ROOT=/home/blucia/cvsandbox/panic
PATH=/home/blucia/.local/bin:/home/blucia/cvsandbox/pin/pin-3.18-98332-gaebd7b1e6-gcc-
linux:/home/blucia/.local/bin/:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/mnt/c/Program
Files/WindowsApps/CanonicalGroupLimited.Ubuntu20.04onWindows_2004.2021.610.0_x64__79rhkp1fndgsc:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System3
2/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/mnt/c/WINDOWS/System32/OpenSSH/:/mnt/c/Program Files (x86)/PDFtk Server/bin/:/mnt/c/Program
Files/MIT/Kerberos/bin:/mnt/c/Program Files/OpenAFS/Common:/mnt/c/Program
Files/OpenAFS/Client/Program:/mnt/c/Users/bluci/AppData/Local/Microsoft/WindowsApps:/snap/bin
HOSTTYPE=x86_64
RISCV=/home/blucia/cvsandbox/panic-riscv-tools/riscv-tools
_=/usr/bin/env
blucia@Nuugaatsiaq:~$
```

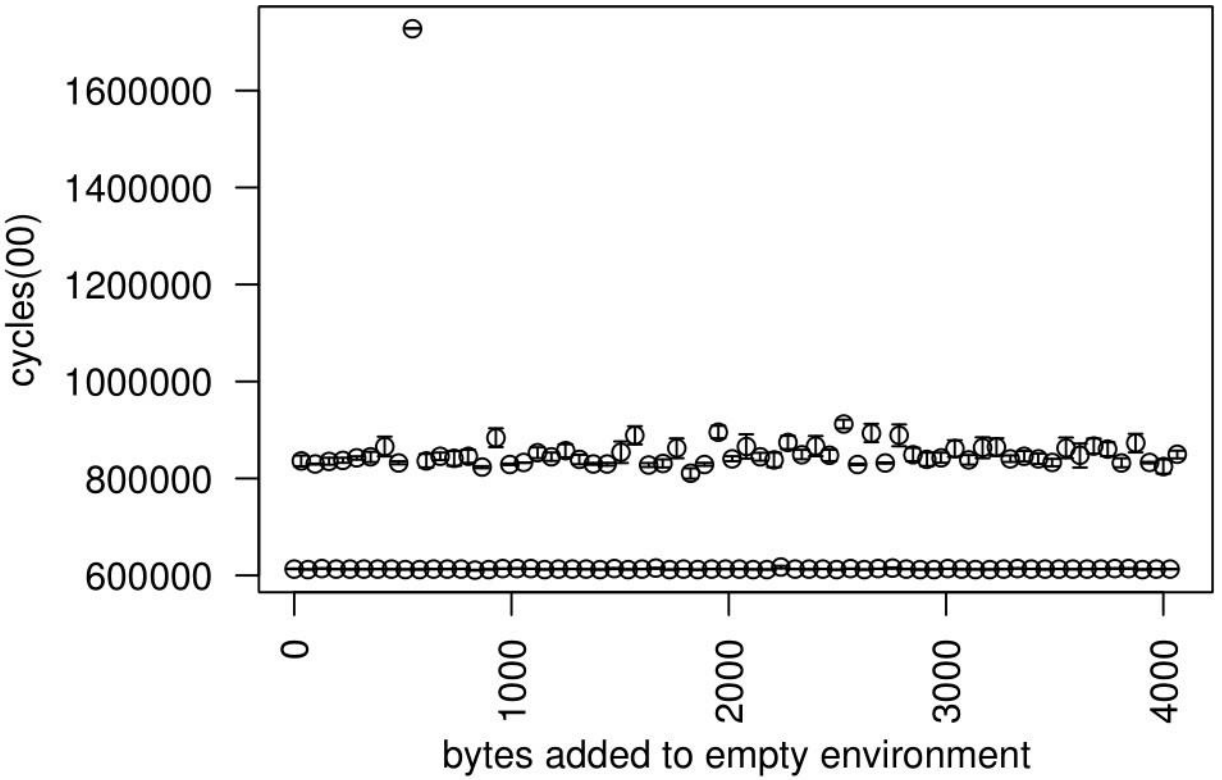
int execvpe(const char *file, char *const argv[], char *const envp[]);

Every process includes an **environment** that takes up some memory in its address space

The size of your Unix environment can bias your measurements!

```
static int i = 0, j = 0, k = 0;

int main() {
  int g = 0, inc = 1;
  for (; g<65536; g++) {
    i += inc;
    j += inc;
    k += inc;
  }
  return 0;
}
```

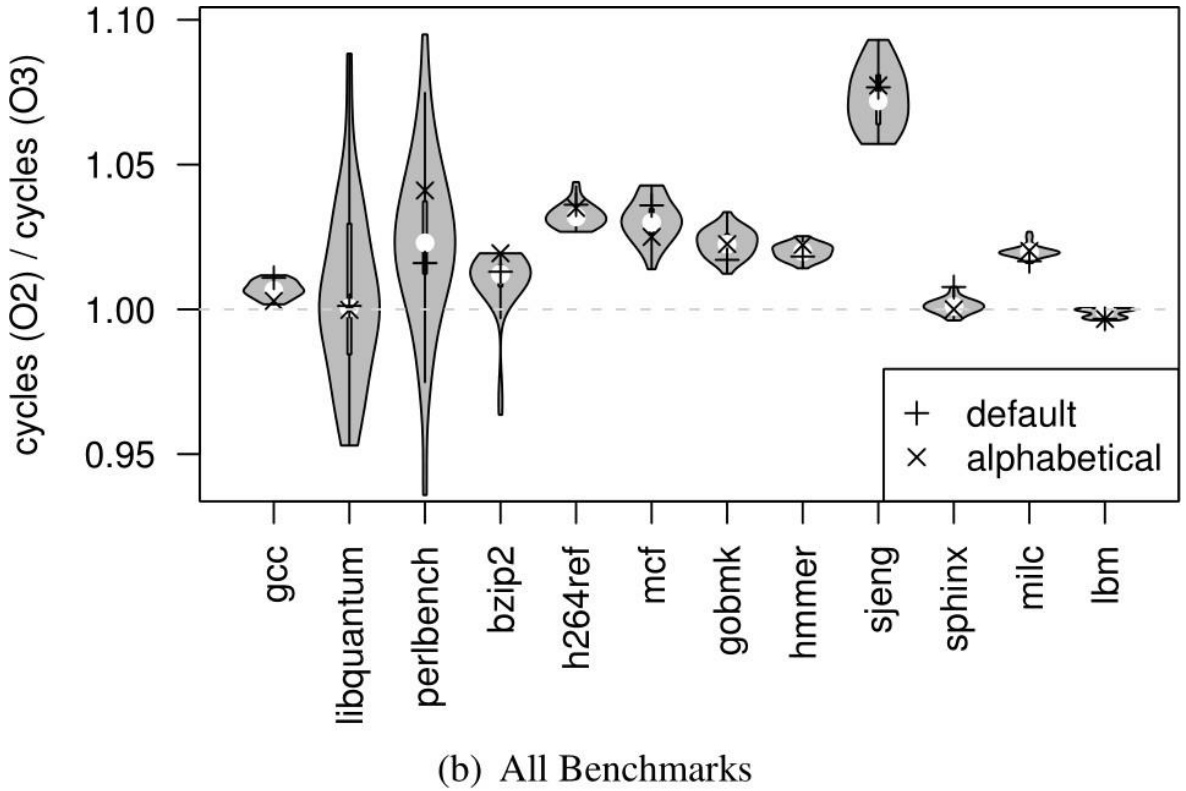
(a) C code for micro-kernel



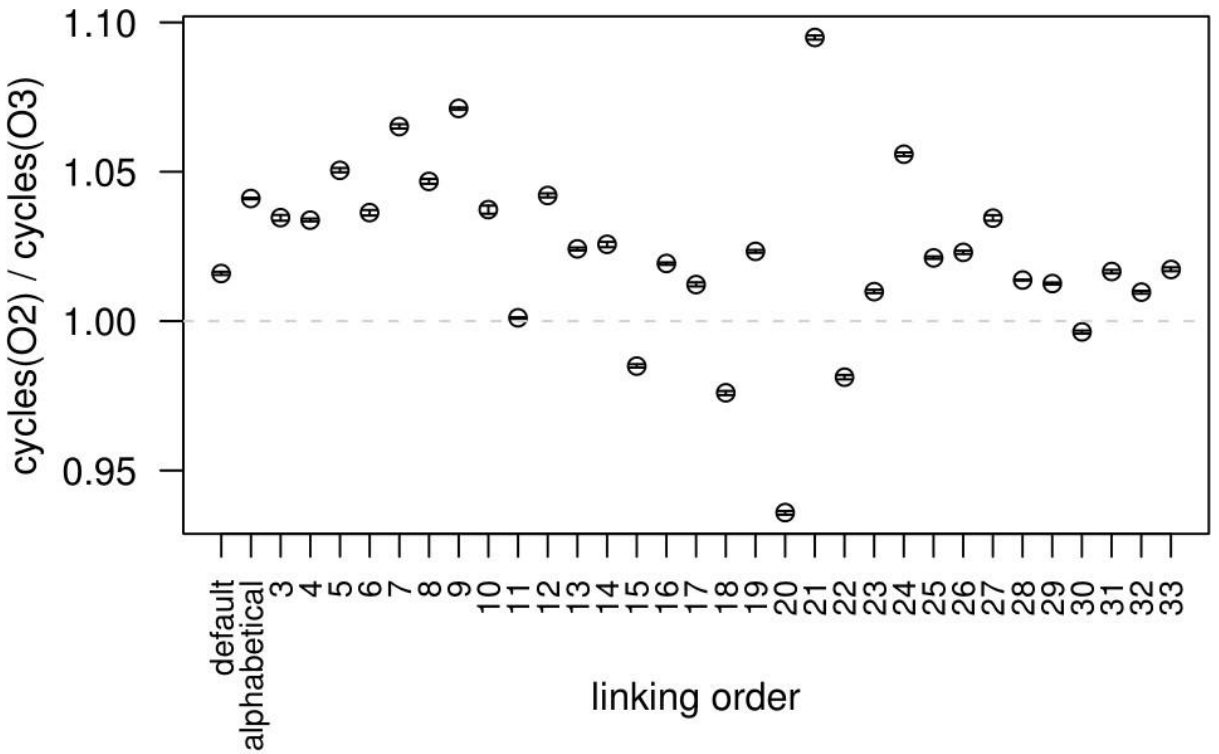
(b) Effect of environment variable size on performance.

(It can change alignment of data on the stack, changing cache conflicts, etc.)

The order in which you link libraries affects optimizability (O2 vs. O3)



(b) All Benchmarks



(a) Perlbench

(Changing locations changes alignment w.r.t. cache conflicts, as very nuanced things such as branch prediction, alignment within buffers/queues, etc.)

Measurement bias effects may exceed the scale of actual effects

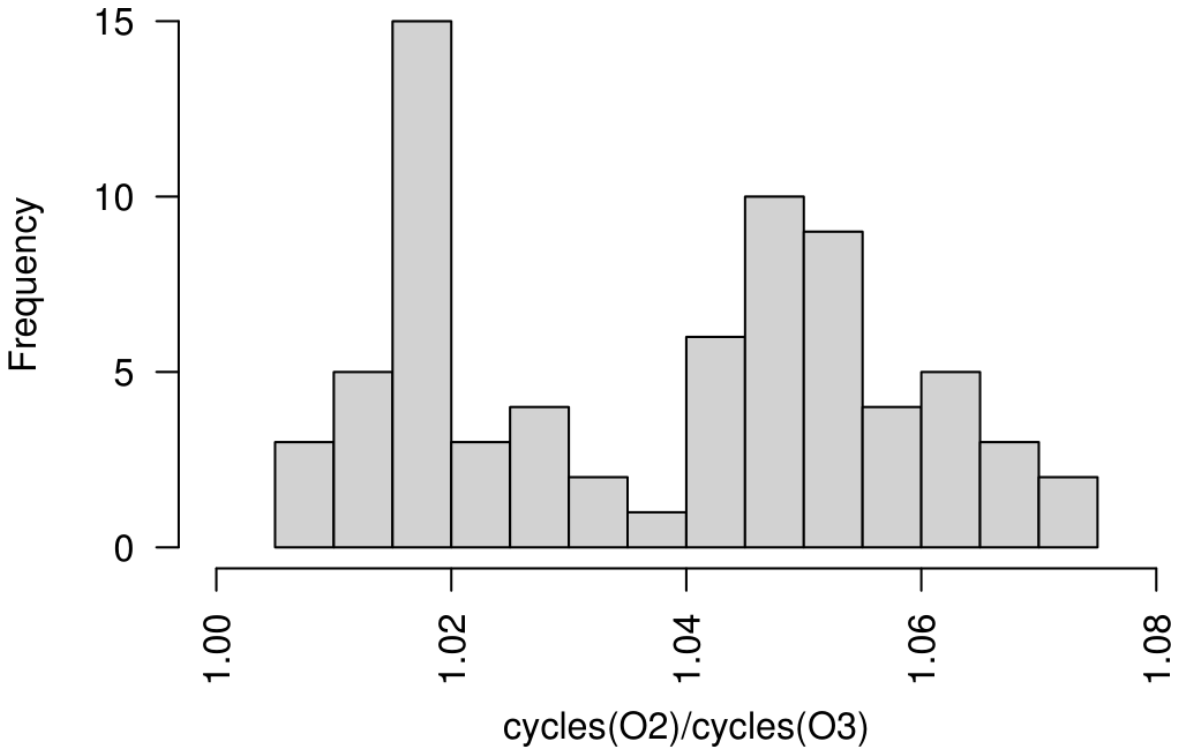


Figure 6. Distribution of speedup due to *O3* as we change the experimental setup.

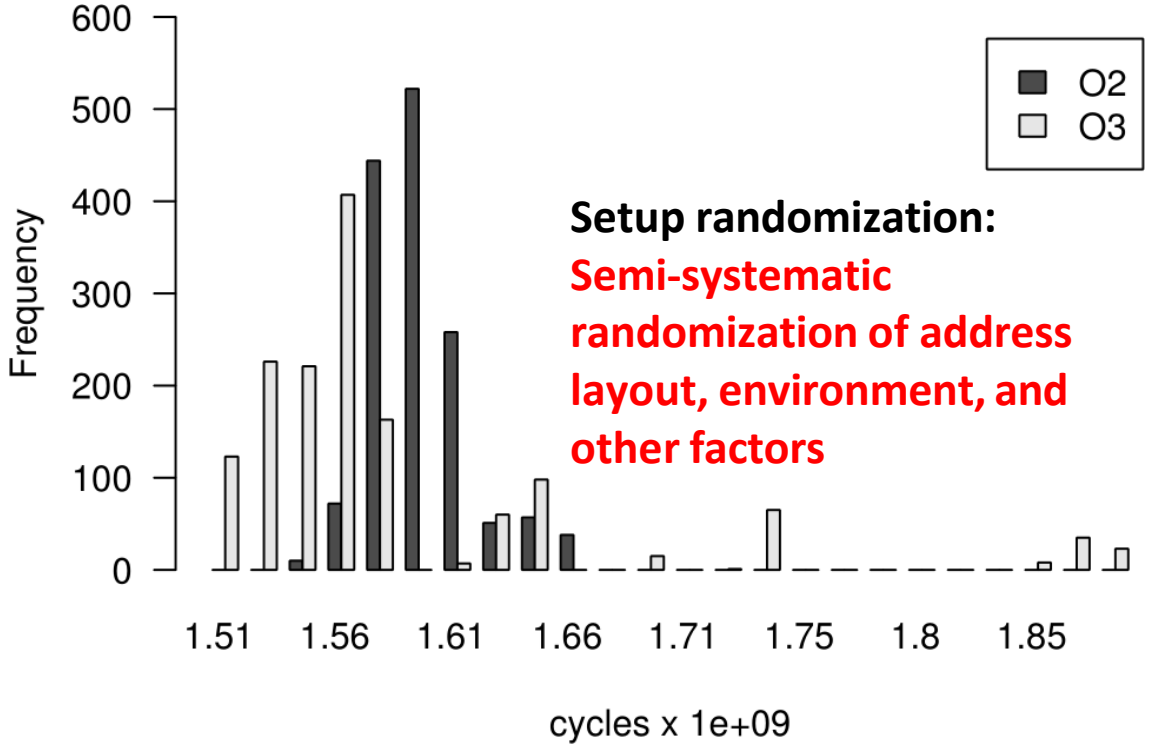


Figure 7. Using setup randomization to determine the speedup of *O3* for perlbench. At a 95% confidence interval, we estimate the speedup to be 1.007 ± 0.003.

It may be impossible to entirely isolate

- Real world benchmarking is most relevant to the real world
- It is also most subject to environmental interference
- One can’t eliminate every aspect of environment, e.g. OS activity
  - Even comparing only user time ignores the impact of side-effects
- It may be that the best one can do is to clean up as much as possible and then exclude outliers.

What did we just learn?

- Performance Evaluation
  - What does it mean to evaluate a system?
  - How to compare two systems?
  - How to do a performance experiment?
  - Measurement pitfalls & metrics
  - Single-number performance summaries
  - Averages
  - Measurement bias sources

What to think about next?

- Performance Evaluation (next time)
  - Design spaces, Pareto Frontiers, and design space exploration
- Miscellaneous (micro)architectural tricks & optimizations (future)
  - Vector processors, SIMD/SIMT, dataflow