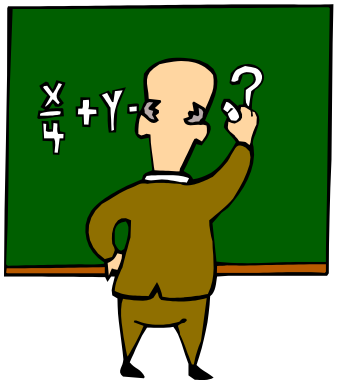
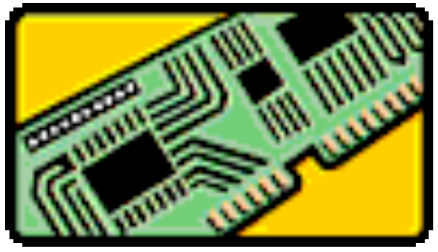


# Fun with Metrics & Statistics



# Problem

- Building actual chips is extremely expensive
- Architecture simulators take a REALLY long time to simulate
- Programs have billions and billions of instructions, so we need to be selective on which we simulate
  - How do you pick?
  - How many do you need to do?
- Which programs do you choose?

# SPEC 2000 CPU Benchmarks

| Benchmark | Language | Category |
|---|---|---|
| 164.gzip | C | Compression |
| 175.vpr | C | FPGA Circuit Placement and Routing |
| 176.gcc | C | C Programming Language Compiler |
| 181.mcf | C | Combinatorial Optimization |
| 186.crafty | C | Game Playing: Chess |
| 197.parser | C | Word Processing |
| 252.eon | C++ | Computer Visualization |
| 253.perlbmk | C | PERL Programming Language |
| 254.gap | C | Group Theory, Interpreter |
| 255.vortex | C | Object-oriented Database |
| 256.bzip2 | C | Compression |
| 300.twolf | C | Place and Route Simulator |

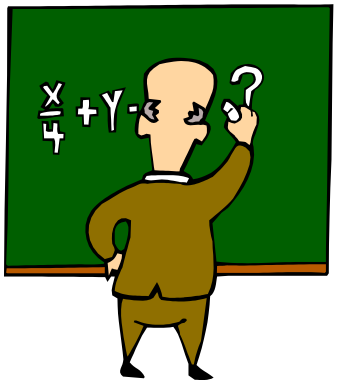
| Benchmark | Language | Category |
|---|---|---|
| 168.wupwise | Fortran 77 | Physics / Quantum Chromodynamics |
| 171.swim | Fortran 77 | Shallow Water Modeling |
| 172.mgrid | Fortran 77 | Multi-grid Solver: 3D Potential Field |
| 173.applu | Fortran 77 | Parabolic / Elliptic Partial Differential Equations |
| 177.mesa | C | 3-D Graphics Library |
| 178.galgel | Fortran 90 | Computational Fluid Dynamics |
| 179.art | C | Image Recognition / Neural Networks |
| 183.equake | C | Seismic Wave Propagation Simulation |
| 187.facerec | Fortran 90 | Image Processing: Face Recognition |
| 188.ammp | C | Computational Chemistry |
| 189.lucas | Fortran 90 | Number Theory / Primality Testing |
| 191.fma3d | Fortran 90 | Finite-element Crash Simulation |
| 200.sixtrack | Fortran 77 | High Energy Nuclear Physics Accelerator Design |
| 301.apsi | Fortran 77 | Meteorology: Pollutant Distribution |

# A Benchmark for Every Purpose

- Parallel processing
  - SPLASH, ParkBench, ScaLapack
- Transactions
  - TPC
- Also benchmarks for
  - Graphics
  - File Systems
  - Networks
  - Web Servers
  - Lots more

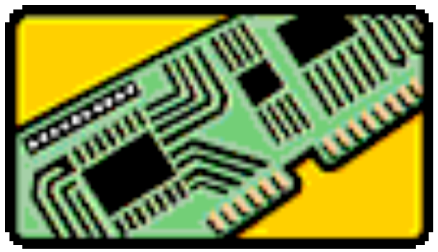
# Metrics

- The key is choosing appropriate metrics to compare your system
- Current important metrics:
  - Performance
  - Power (battery life, reliability)
  - Area (~cost)
  - Complexity (~time to market)
- The metrics are not yet standardized
- Even proper choice of metrics does not guarentee meaningful results

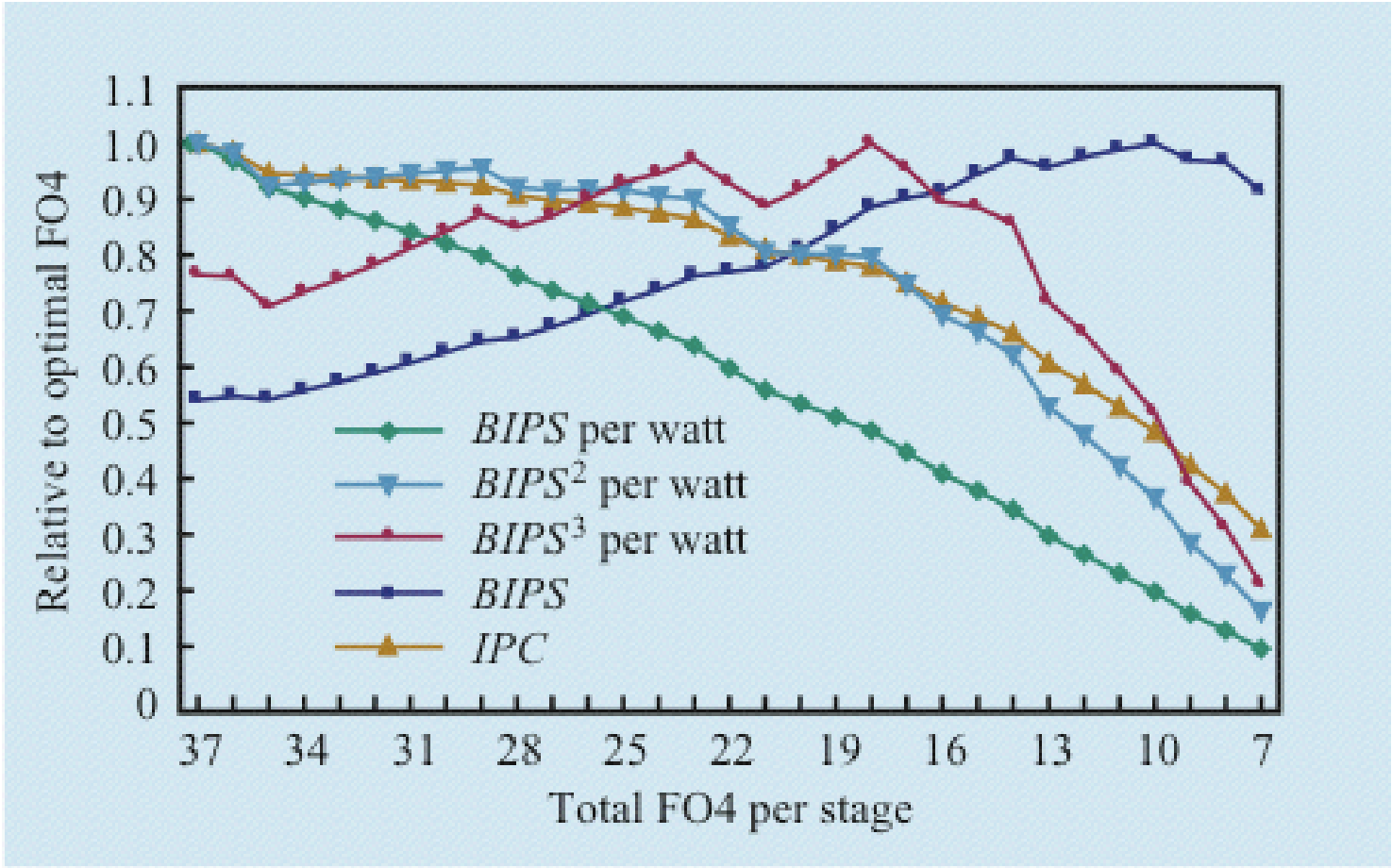


# Fun with Metrics

## ISCA-quality Metrics

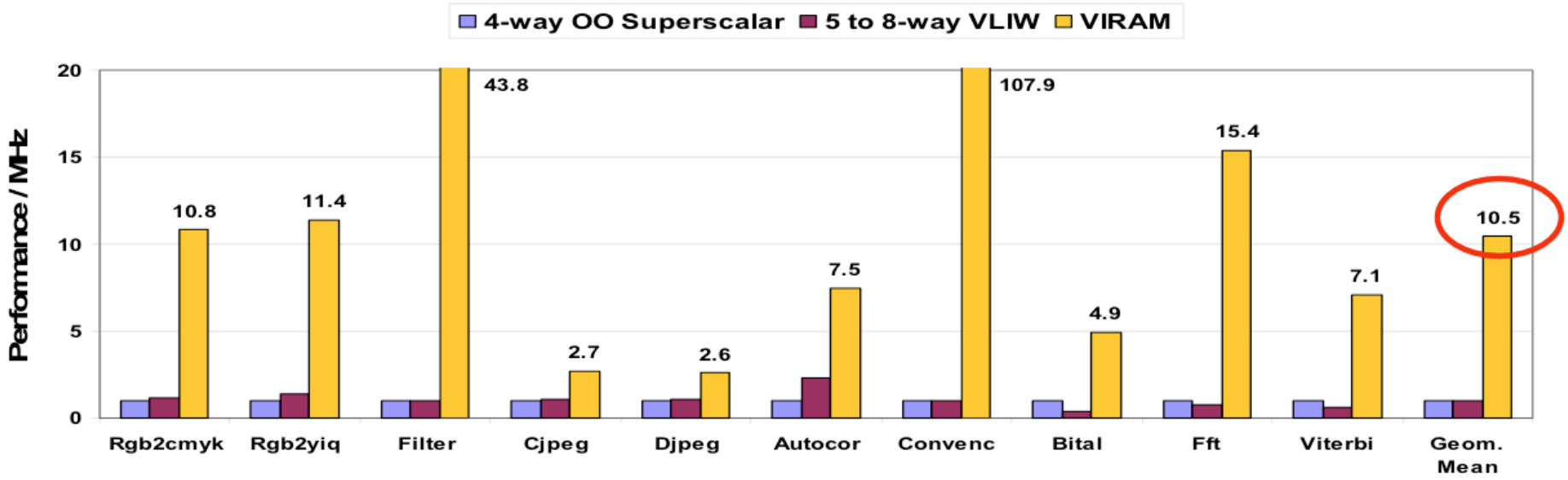


# How to Choose a Metric

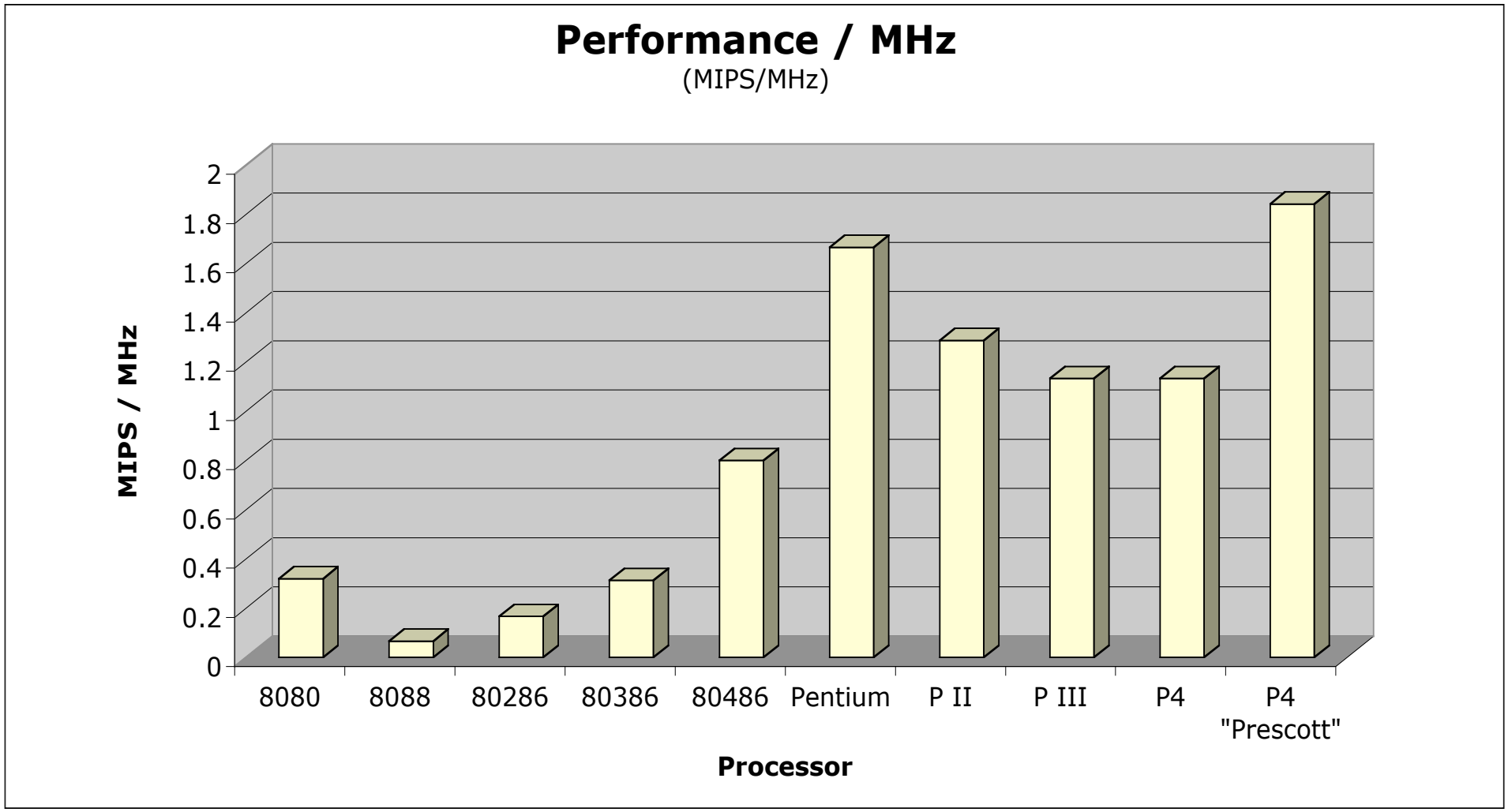



-Only $BIPS^3$ per watt shows an optimum point that is not at the shallowest pipeline depth

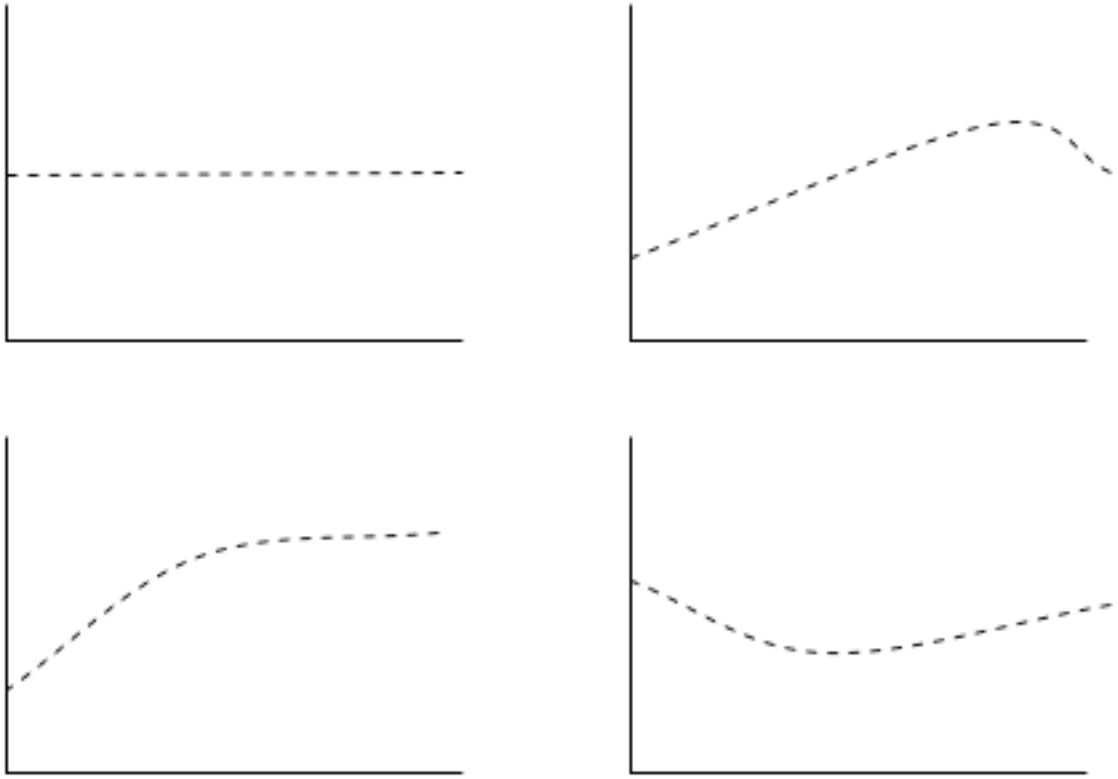
# VIRAM Performance (per MHz)



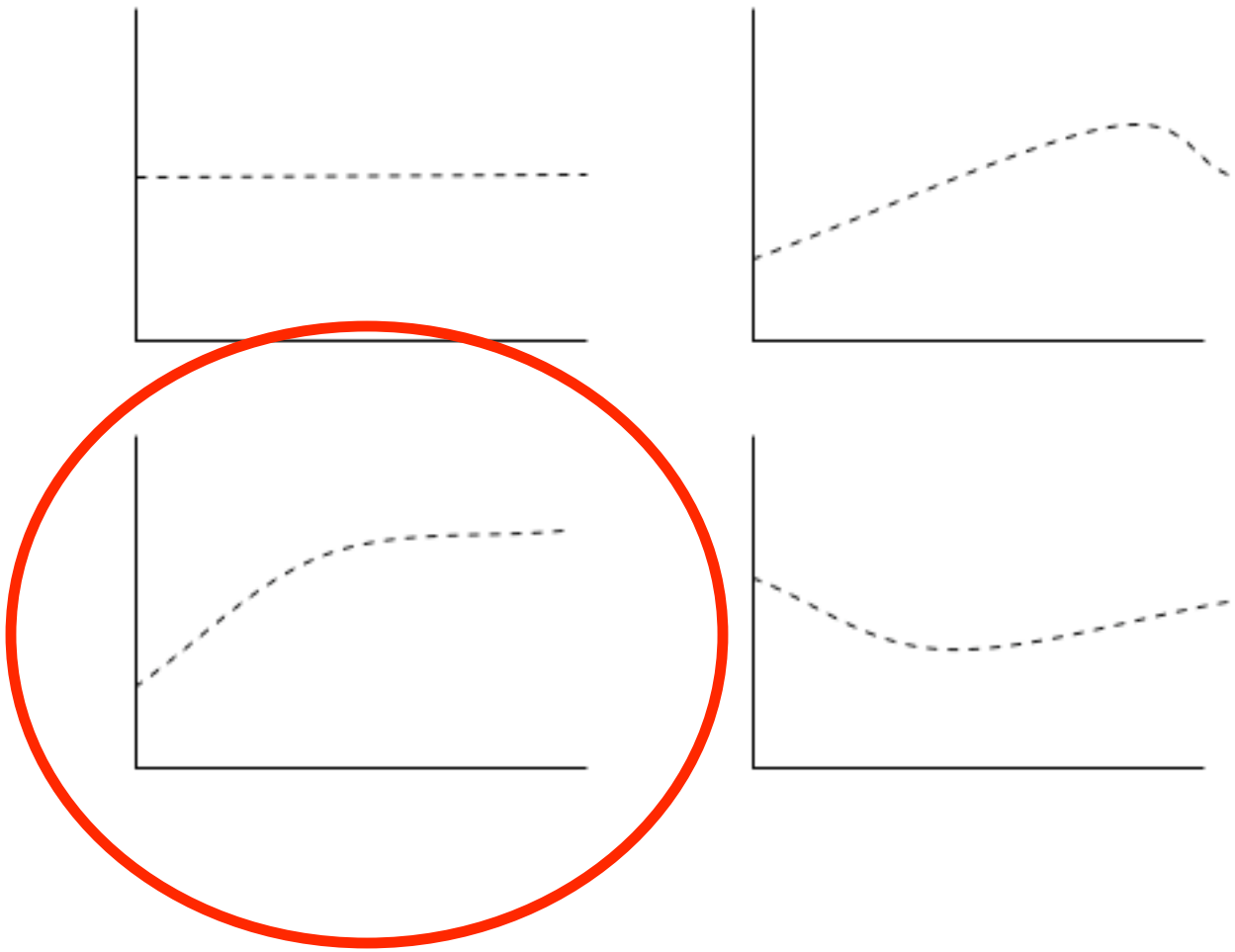
# Helpful Performance Metrics



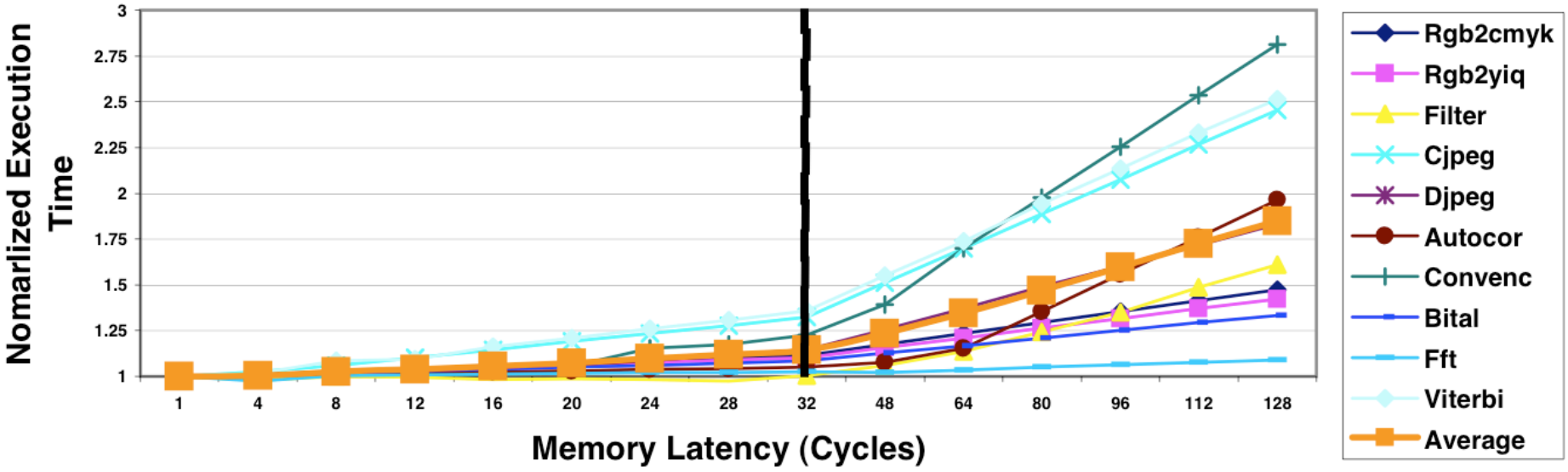
# Who remembers this assignment?



# Who remembers this assignment?

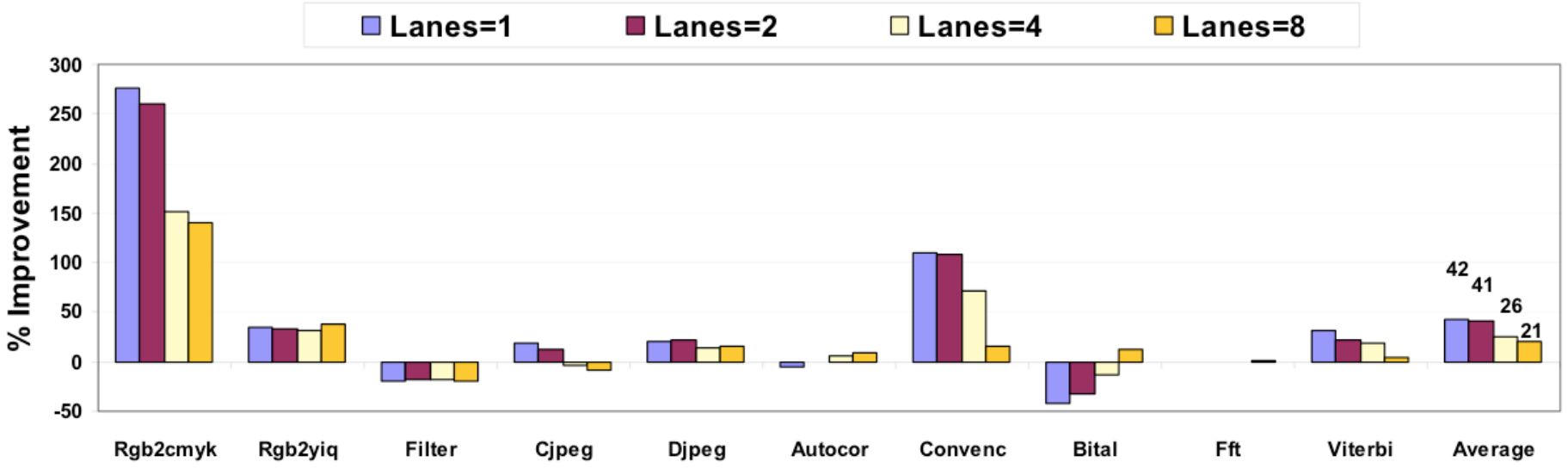


# How to make a knee in the graph



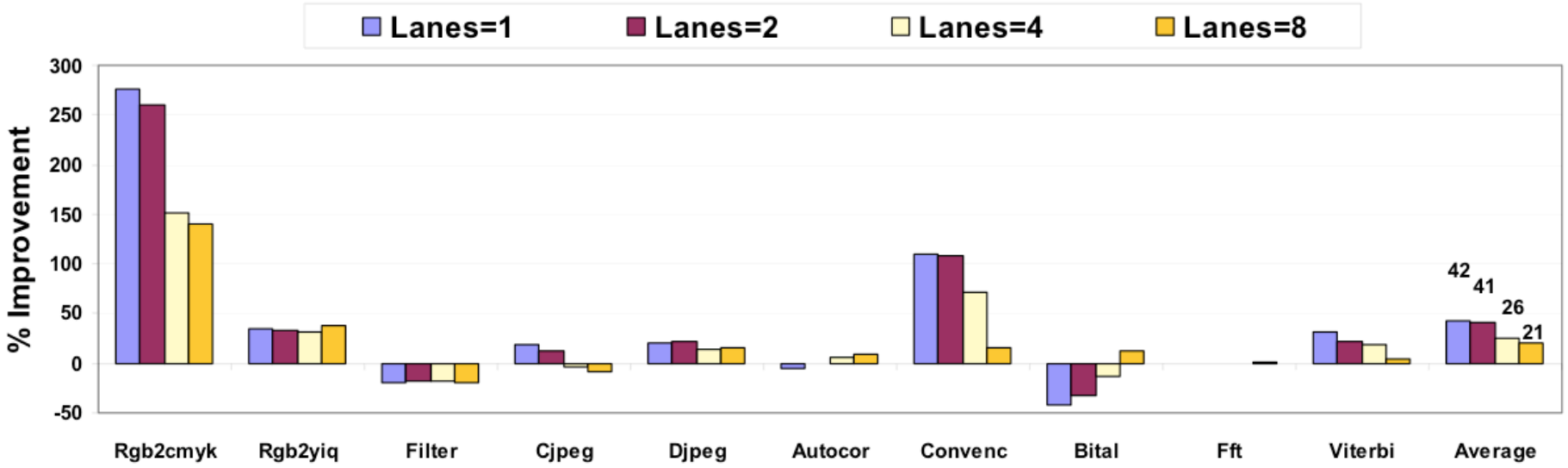

3 4 4 4 4 4 4 4 16 16 16 16 16 16

# 20% on "Average"




- CODE is 20% faster than VIRAM
  - Even for multi-lane implementation of both approaches

# Fun with outlying data points




With Rgb2cmyk

Arithmetic Mean: 21.5%

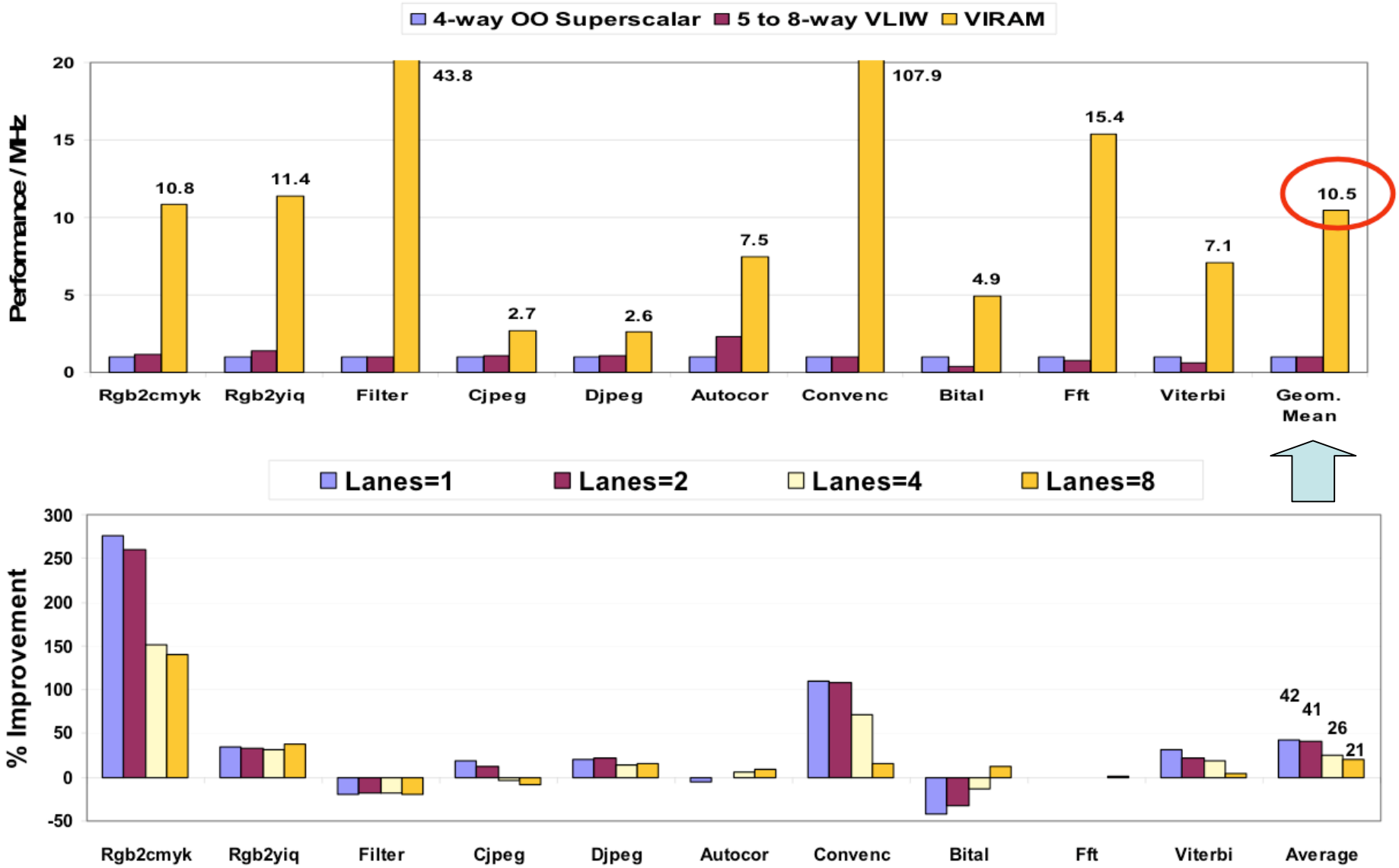
Geometric Mean: 16.1%

Without Rgb2cmyk

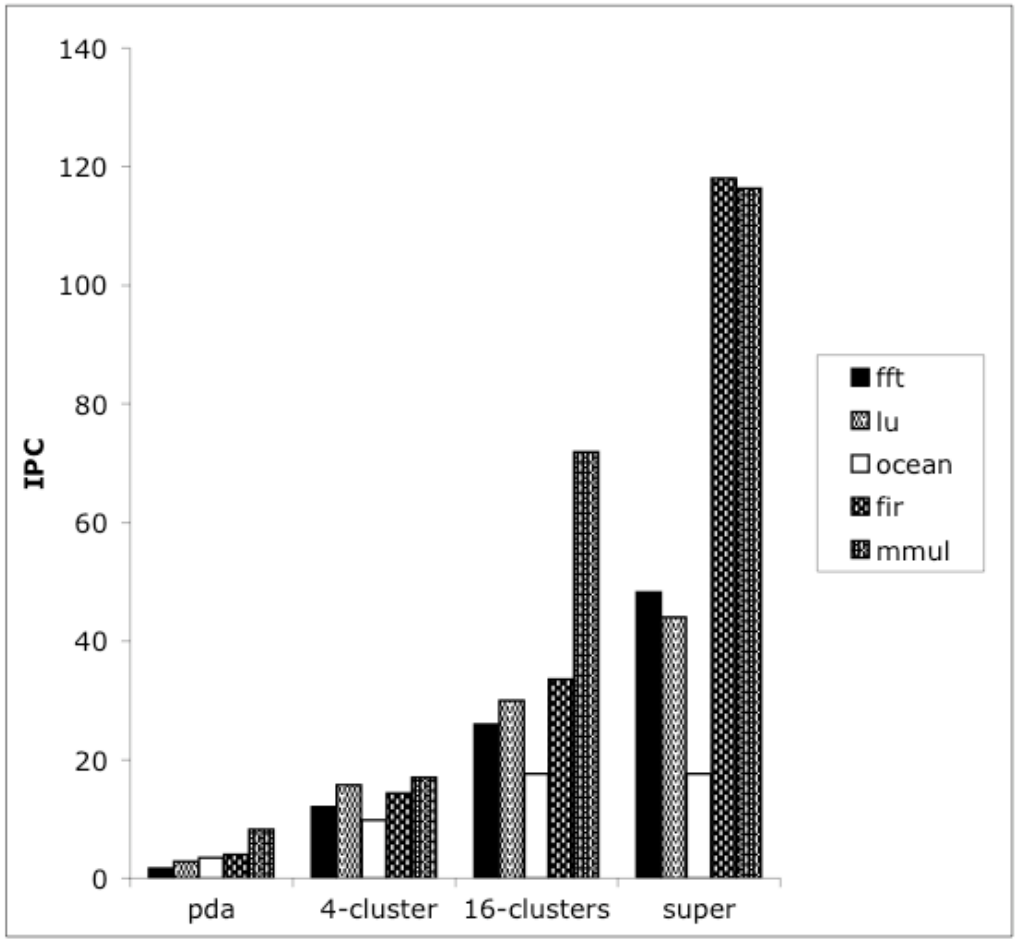
Arithmetic Mean: 8.3%

Geometric Mean: 7.1%

# Average & Geometric Mean Same Thing….



# Improving WaveScalar IPC



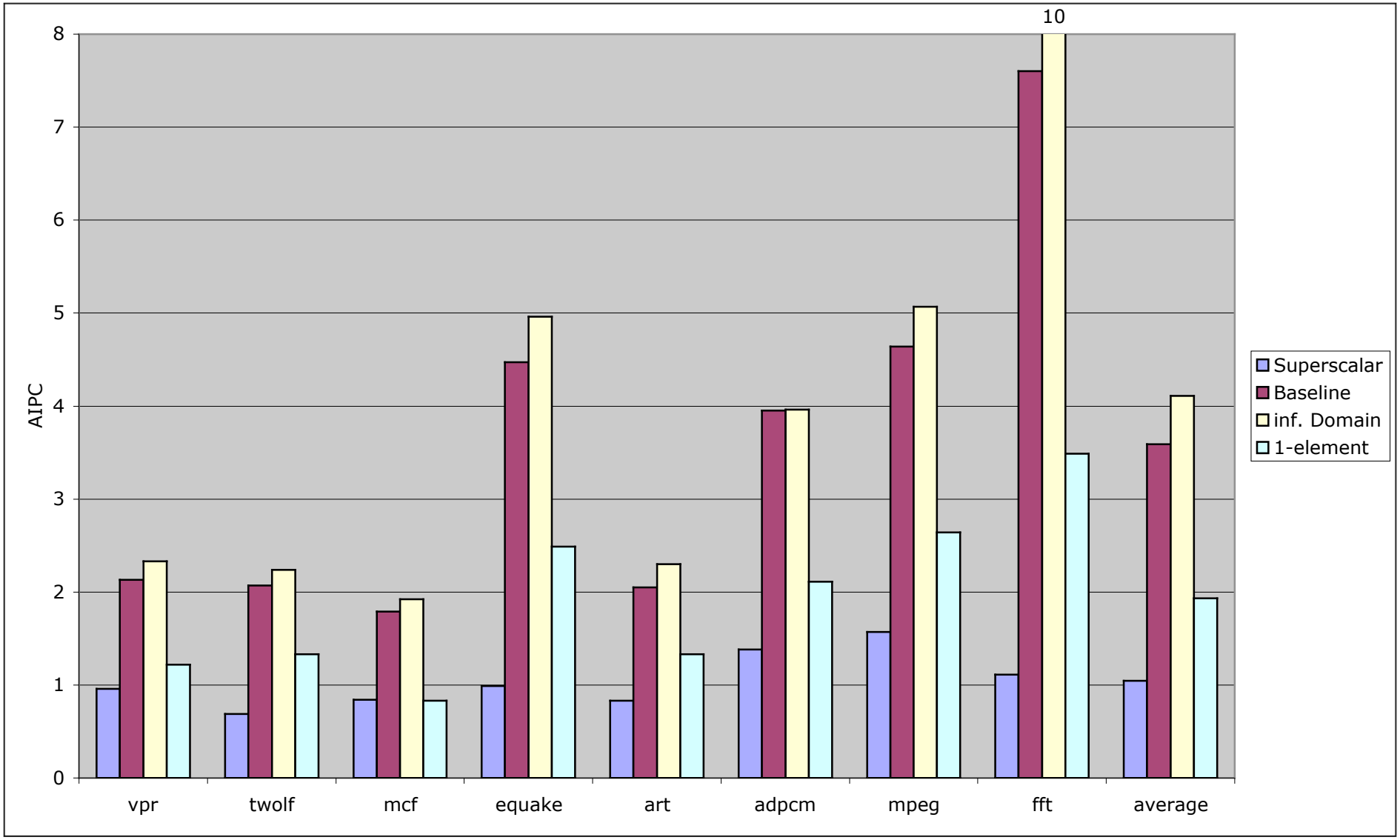

16 Cluster:

Average IPC: 33.8
Geometric: 29.75
Statistical Gain: 13.6%

Super:

Average IPC: 68.8
Geometric: 55.41
Statistical Gain: 24.2%

# WaveScalar Performance



# “Adjusted” WaveScalar Performance