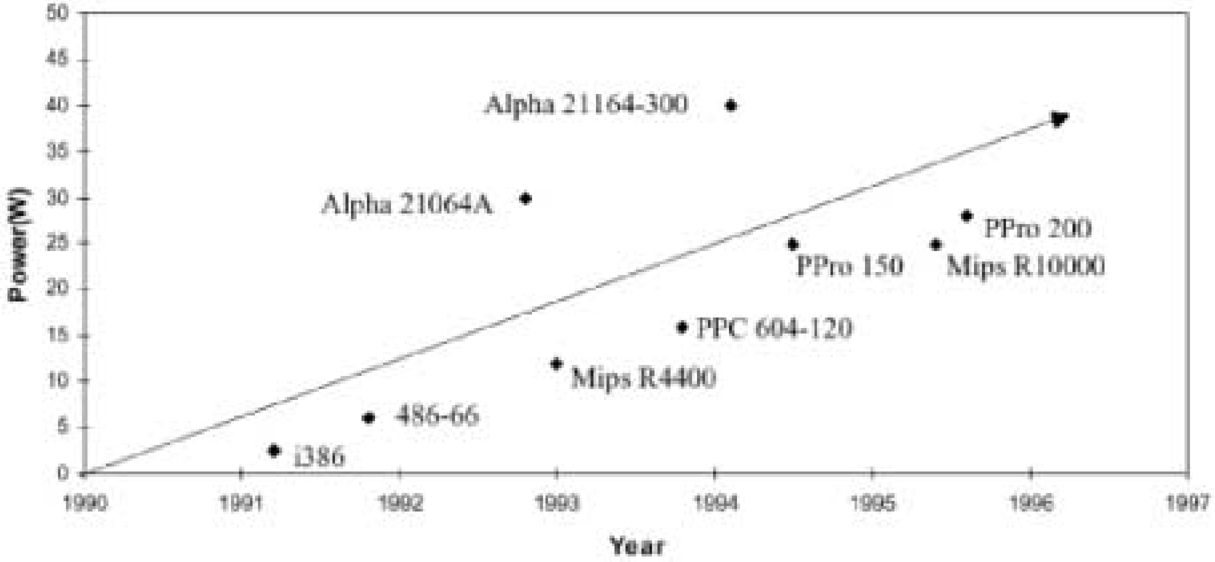
Power-efficient caching

Michael Brogioli

Bryan Jones

Hot, hot, hot…

- Power / mm^2 of die area increasing!
  - Huge heat sinks – liquid cooling, anyone?
- Total power consumption increasing.
  - Reduces rack density – cooling costs $.
  - Mobile applications need not apply.



Cache: the jackpot

- Caches consume:
  - Much of the die.
  - Much of the power (50% - 60% in sims).
- Idea: fit cache size to working set of program.
  - Multimedia apps doesn’t need a data cache!
  - Working sets vary dramatically.
- Results: saved 30% power.

Outline

- Introduction
- Saving power
  - Static / dynamic analysis
  - Simulation methodology
- Results
- Conclusions

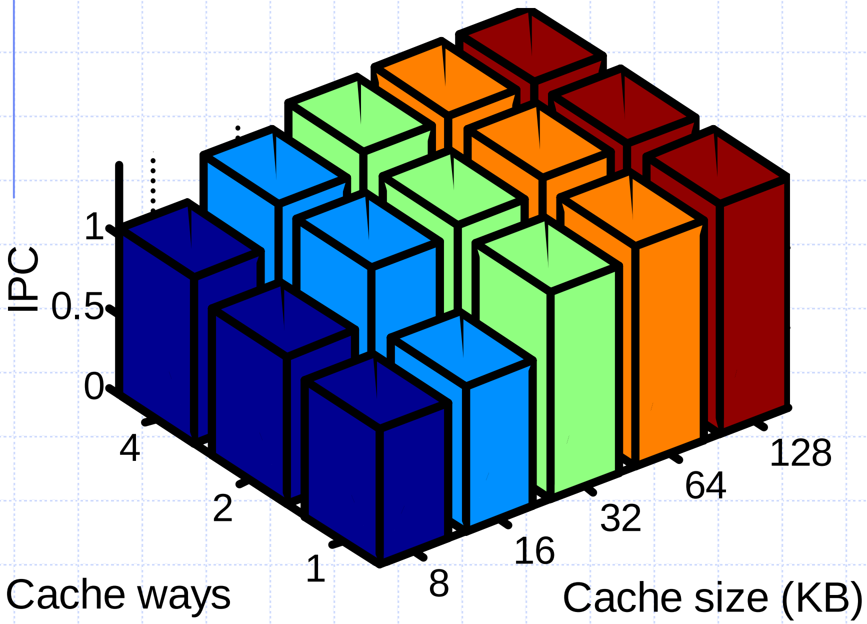
Two ways of watching Watts

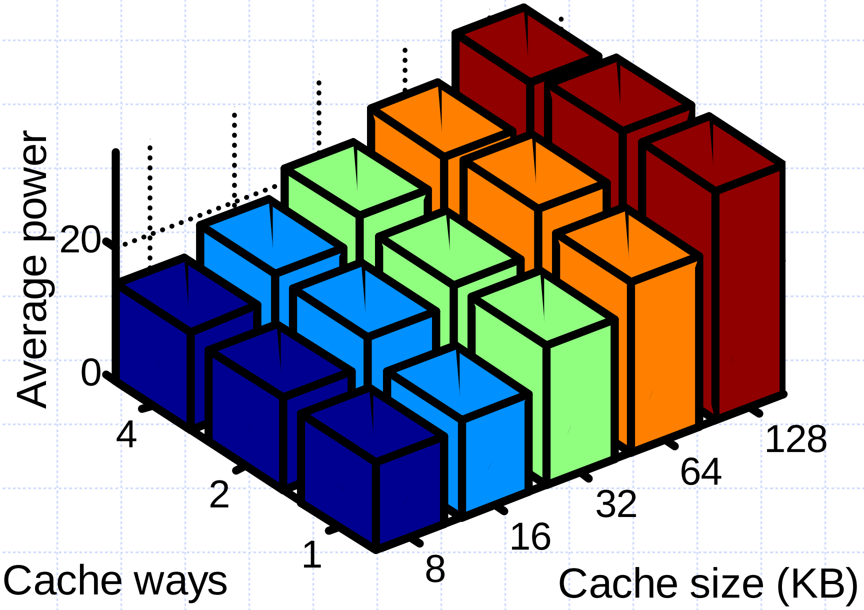
- What’s a Watt?
  - Power = Energy per unit time.
  - “Power” means average power.

1. Static analysis: for each program, choose a “best” cache size.
2. Dynamic analysis: as a program runs, fit cache size to the current working set.

Static analysis

- Vary cache size and associativity. Measure IPC and average power.

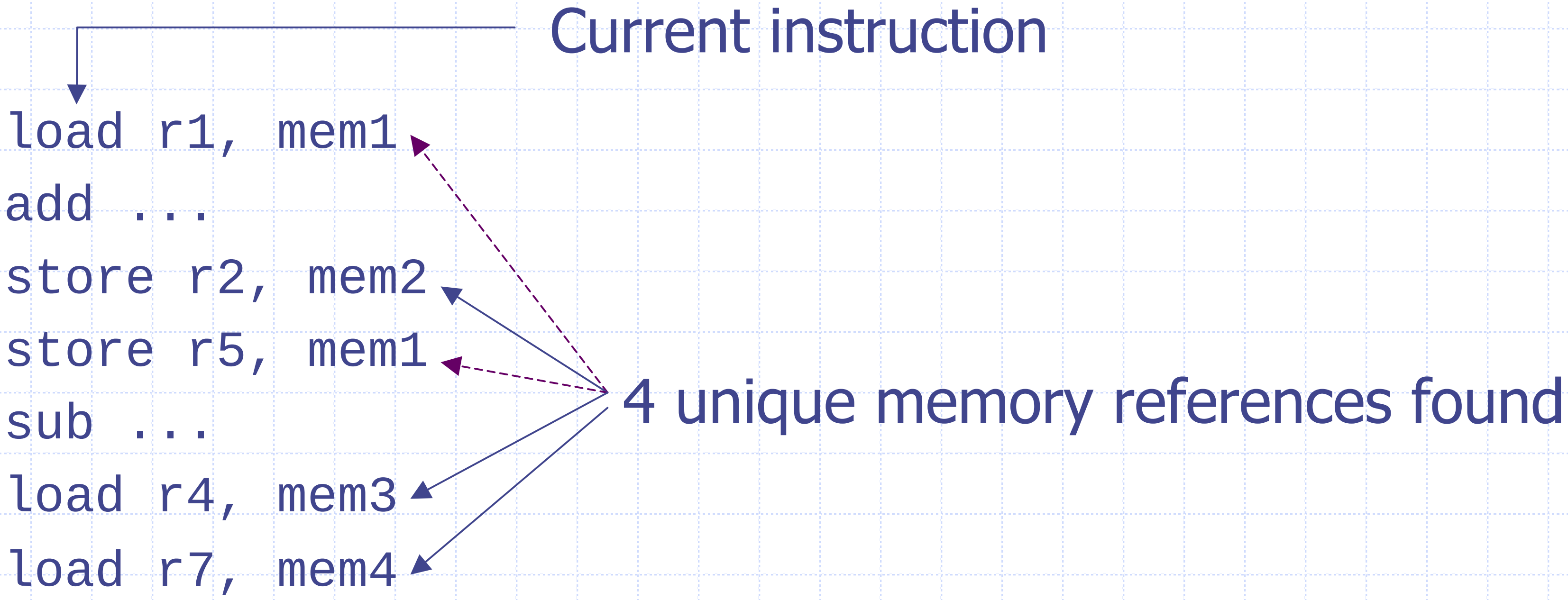




- Find a good compromise.

Dynamic analysis in 3 steps

1. Count the number of unique memory references made in the future.



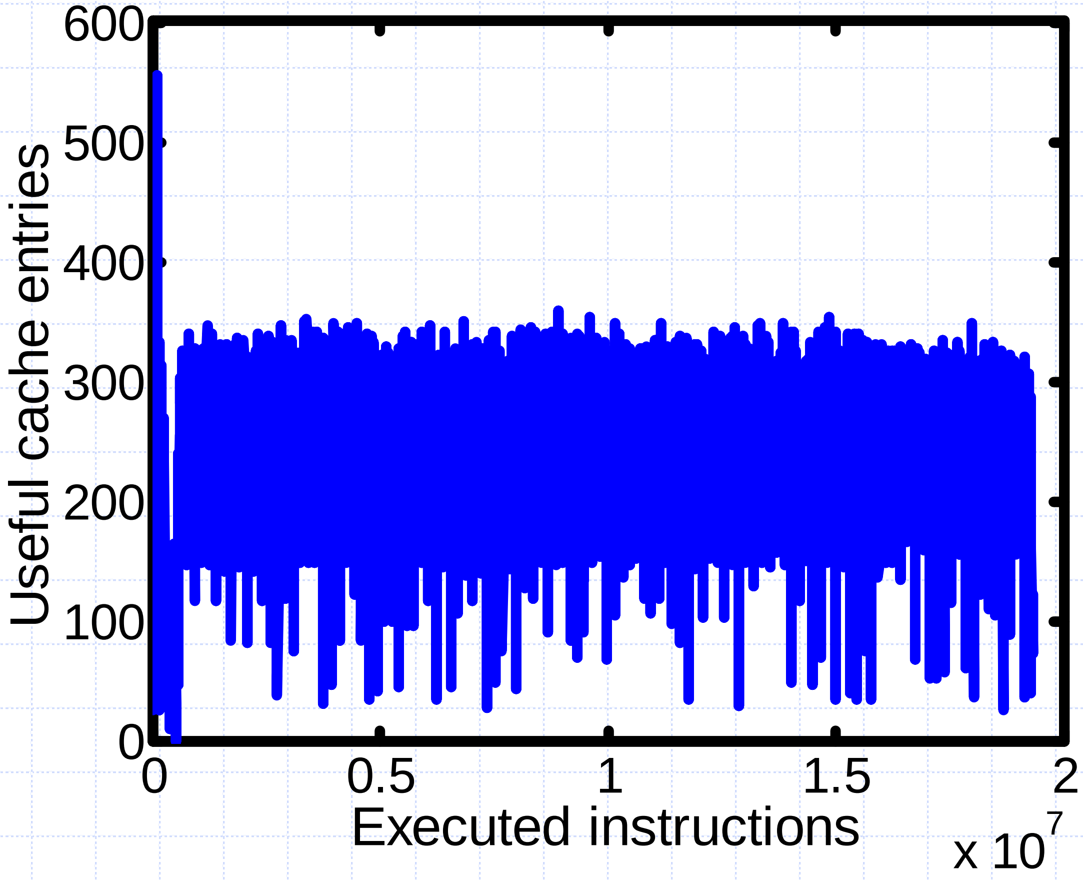
Current instruction

```
load r1, mem1
add ...
store r2, mem2
store r5, mem1
sub ...
load r4, mem3
load r7, mem4
```

4 unique memory references found

Dynamic analysis in 3 steps

2. Plot number of references against time.



3. Size cache accordingly.

Power simulation

- Wattch method:
  - Model power consumption in a processor.
  - Each cycle, bill each utilized unit of the processor for power used.
- Tools:
  - SimpleScalar v3.0 (supports PISA).
  - Wattch v1.02 integrated in.
  - Added features for dynamic analysis.

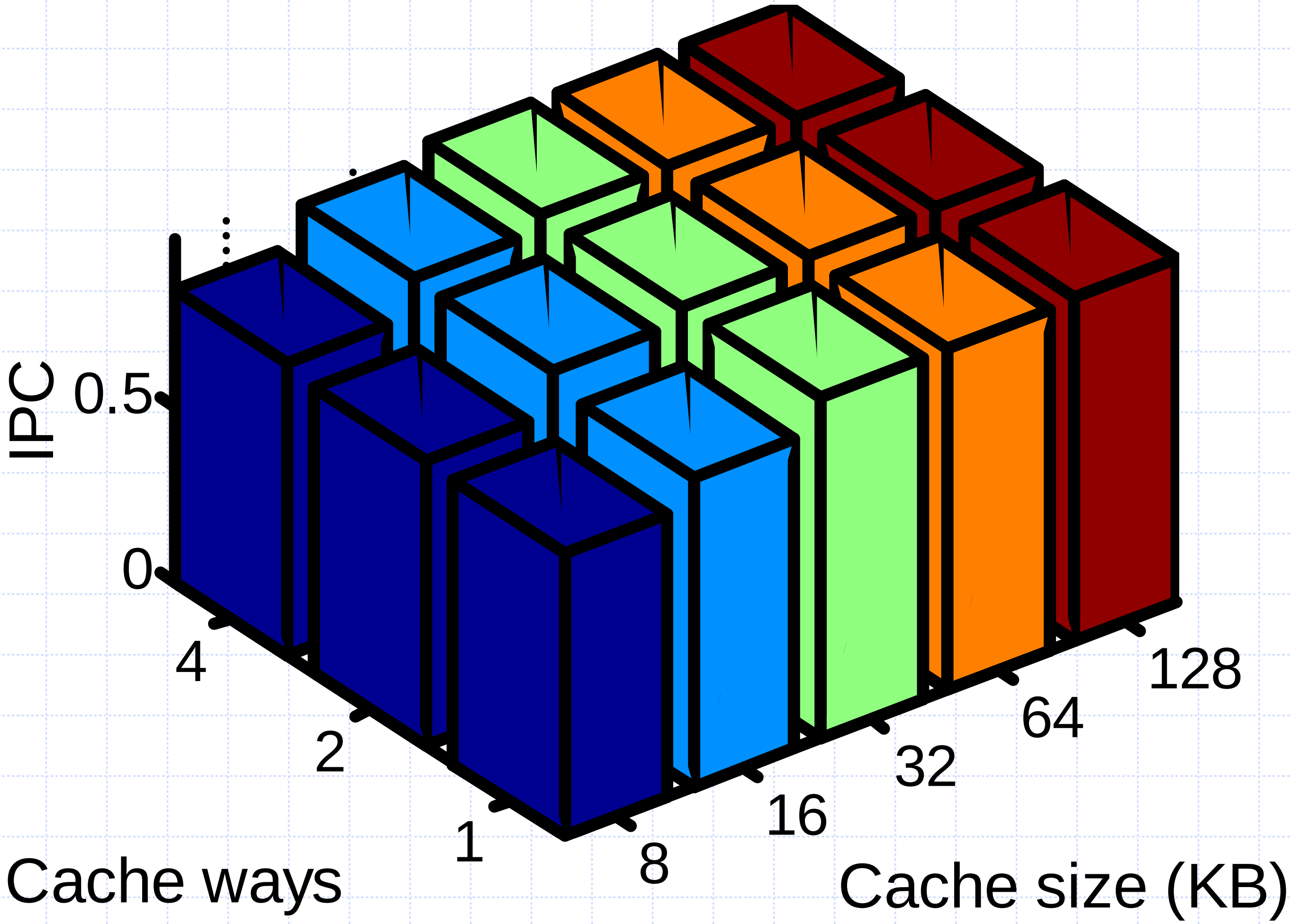
Processor parameters

Cache (I/D same size)	8K-128K, 1/2/4 way, 32 byte lines
Process technology	0.35 μm, 600 MHz
Clocking	Aggressive conditional
Functional units	5 int, 5 fp, 2 memory
Issue / commit width	4
Register update unit	16 entries
Load/store queue	8 entries
Branch predictor	Bimodal, 4K entries
TLBs	4K entries, 4-way, LRU

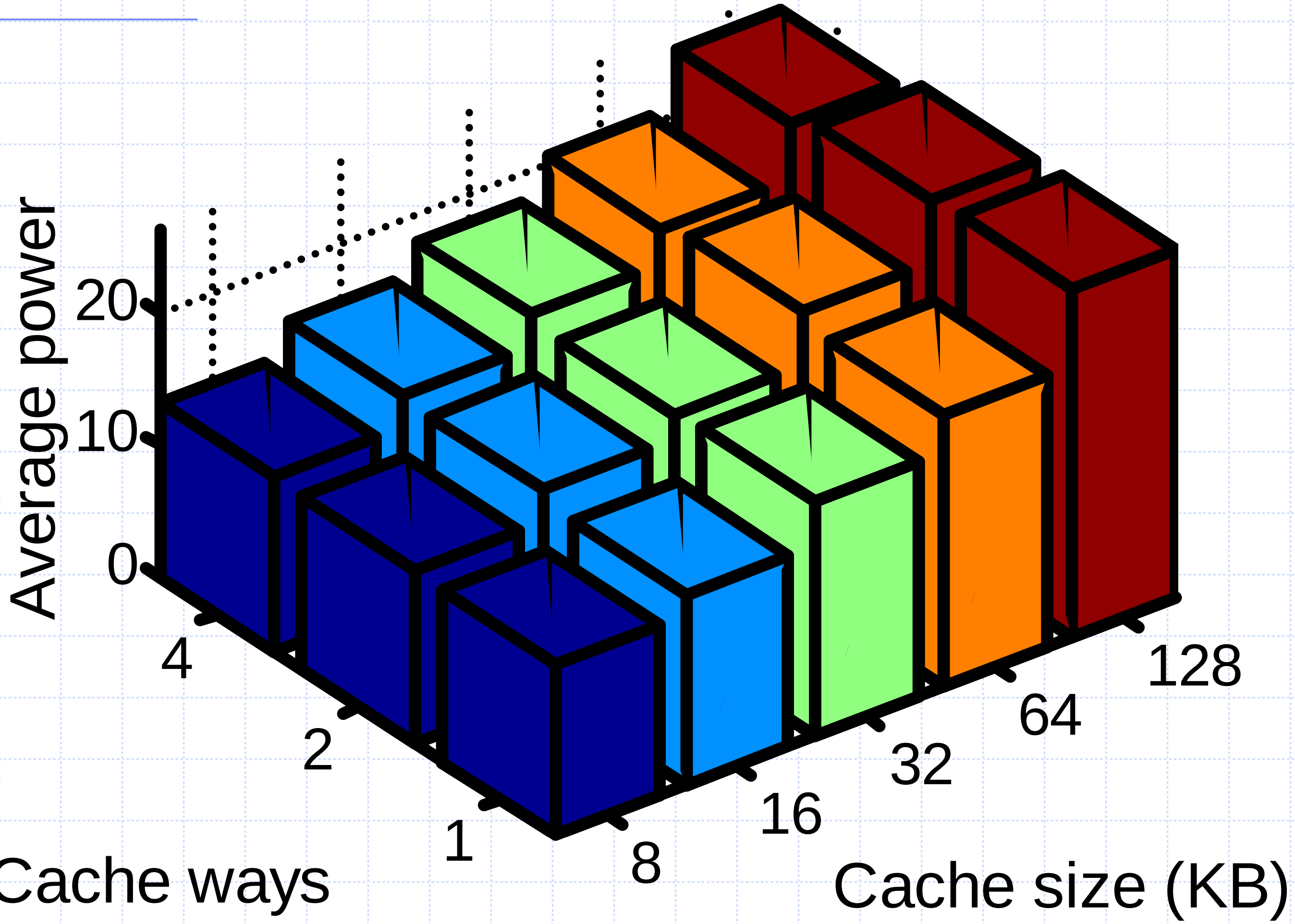
Outline

- Introduction
- Saving power
  - Static / dynamic analysis
  - Simulation methodology
- Results
- Conclusions

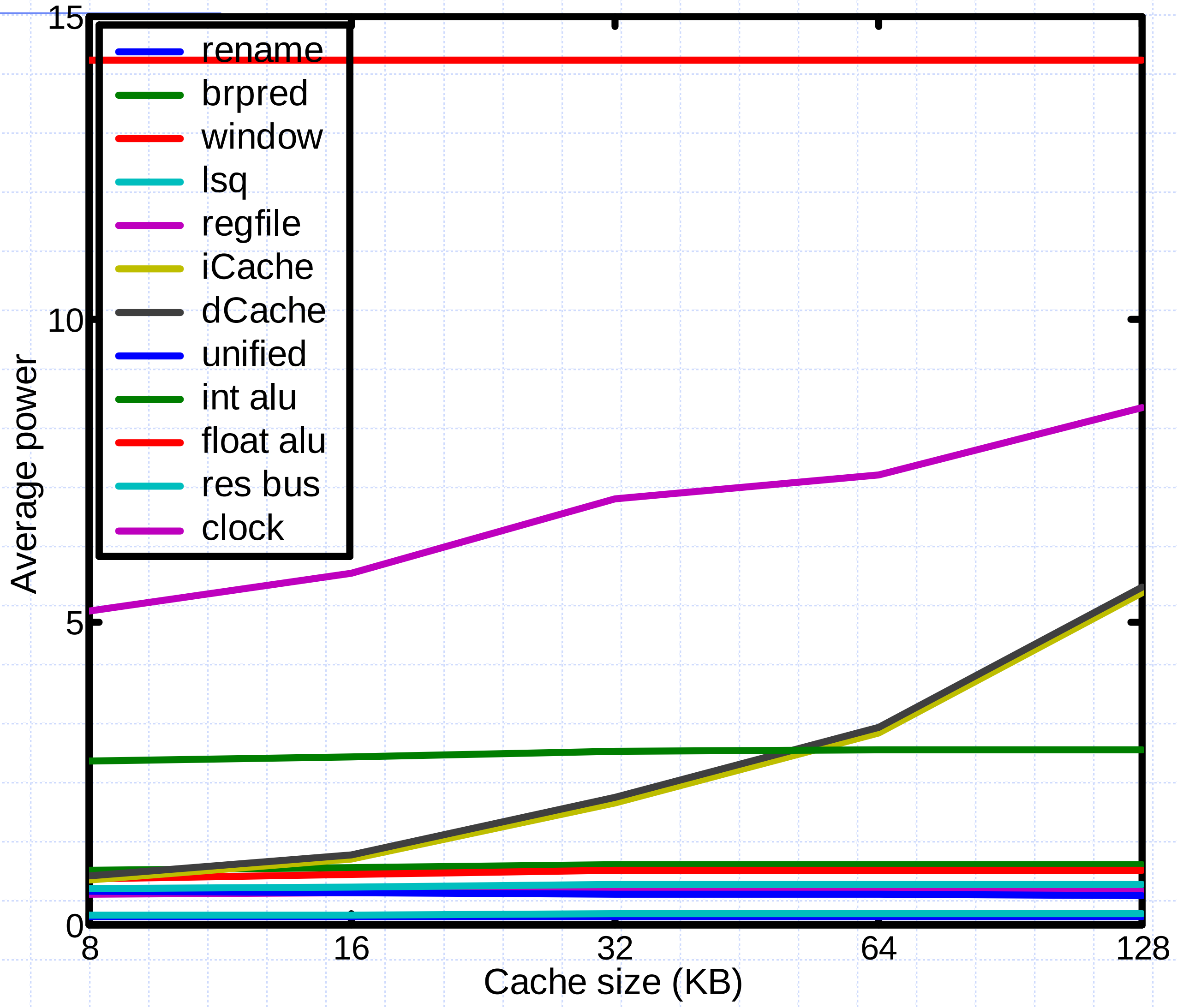
Static analysis of ammp



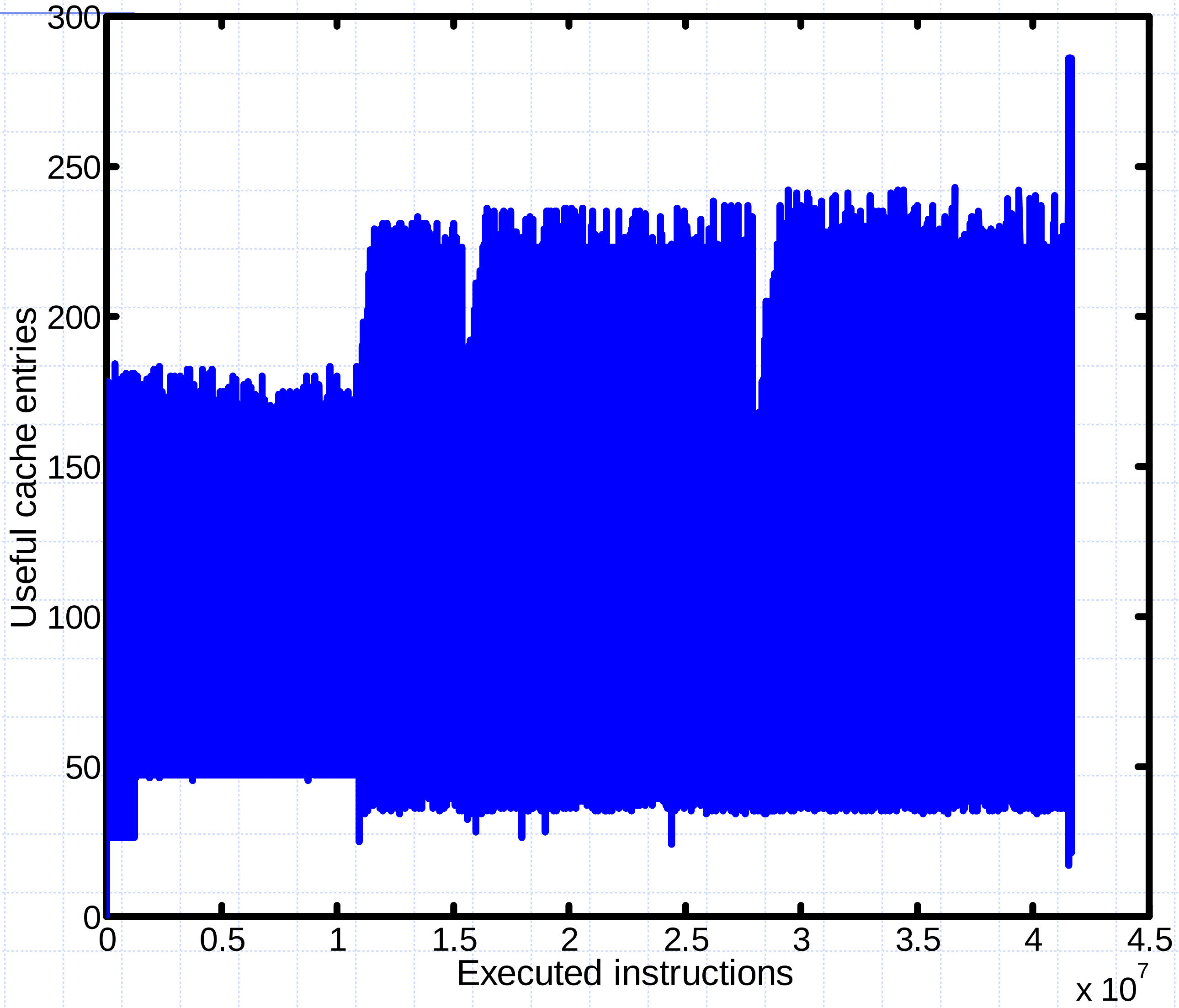
Static analysis of ammp



Static analysis of ammp



Dynamic analysis of ammp



Power savings

- VPR (hardest case):
  - Reduce power by 30%
  - Reduce performance by 10%

  -or-
  - Reduce power by 66%
  - Reduce performance by 50%

Outline

- Introduction
- Saving power
  - Static / dynamic analysis
  - Simulation methodology
- Results
- Conclusions

Conclusions

- Static analysis:
  - An informed tradeoff can be made between power and performance.
- Dynamic analysis:
  - Analysis demonstrates fine-grained power savings is available.

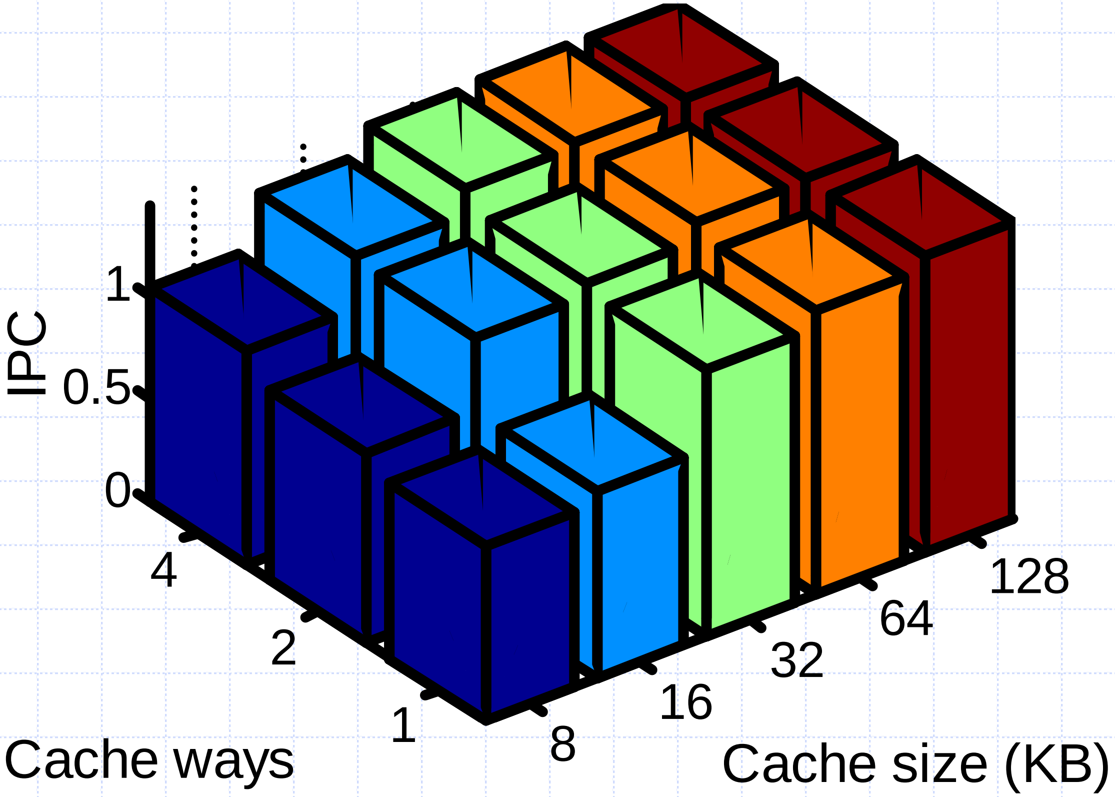
Hindsight

- Split i-cache sizing from d-cache sizing to make analysis more clear.
- Fully associative caching proved buggy at last minute.
- More analysis preferable.
- Find better comparison metrics.

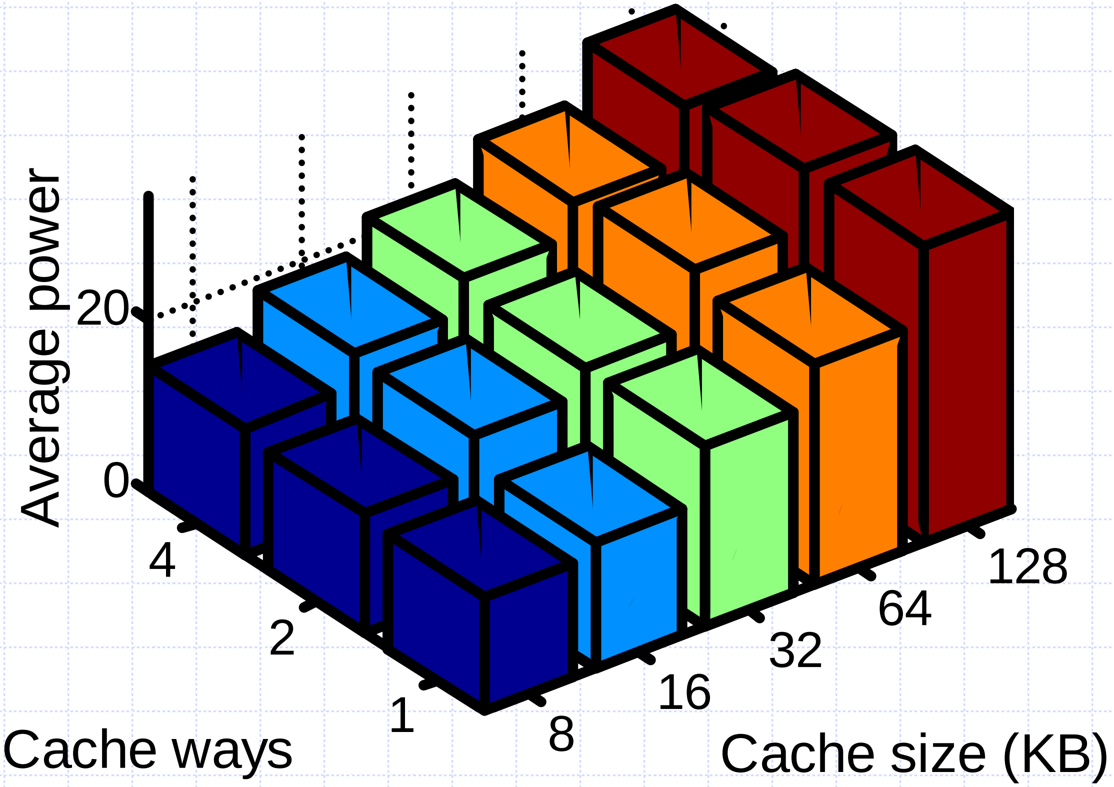
Open questions

- How to map dynamic analysis into cache size?
- How to incorporate either analysis into compiler and hardware?
- What architecture should be used for changing cache size?
- Analyze i-cache behavior and tune.

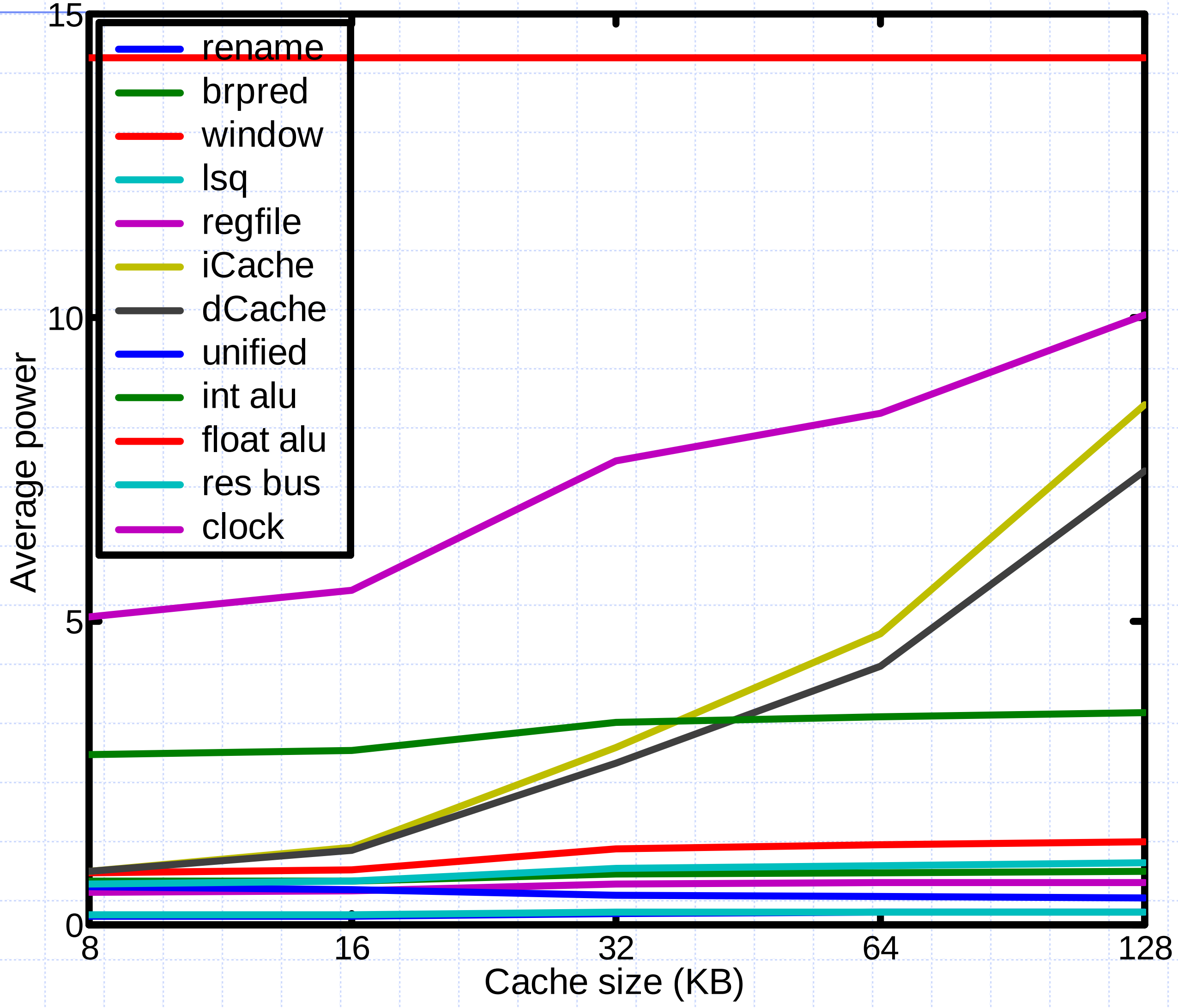
vpr



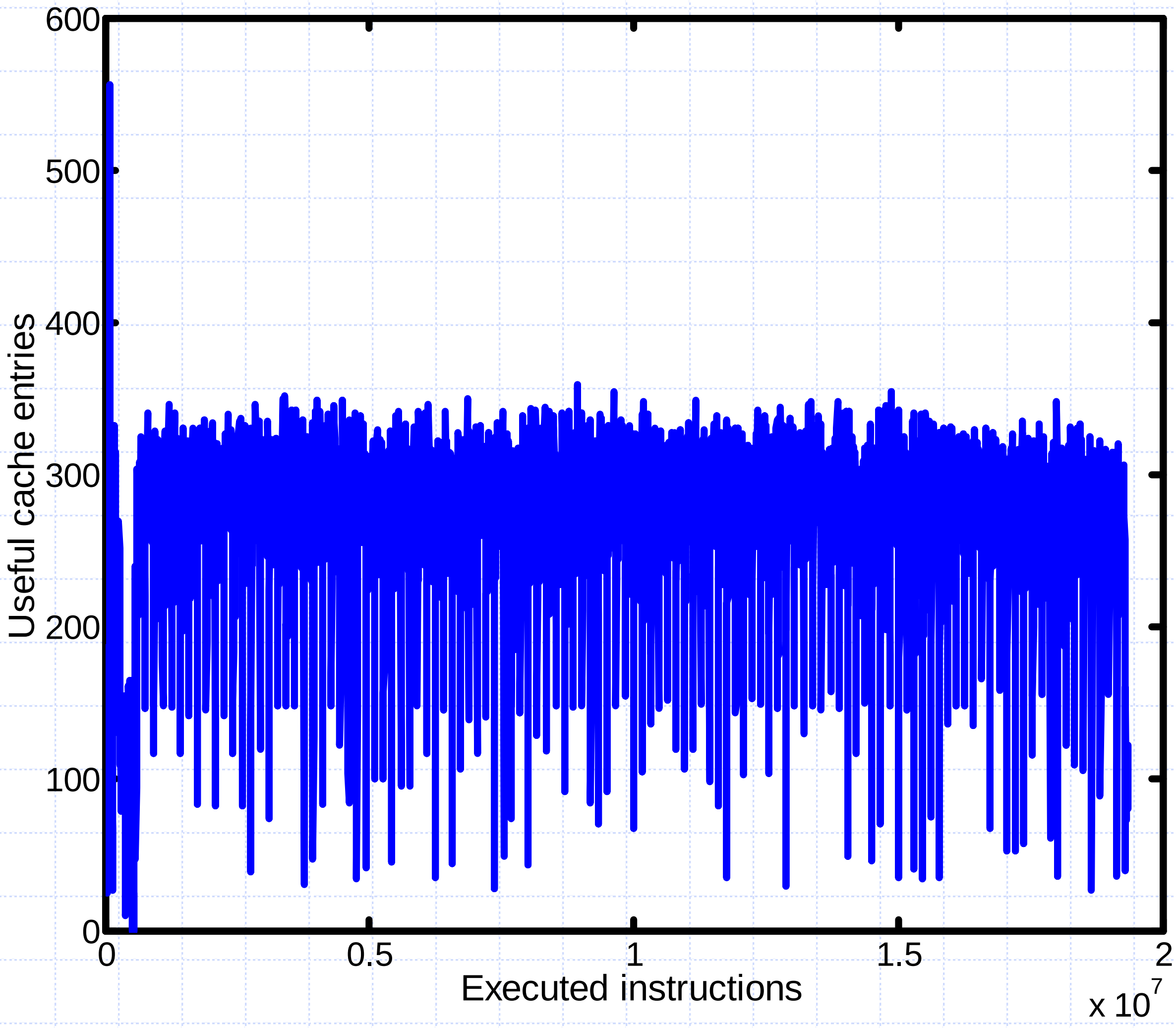
vpr



vpr


rename
brpred
window
lsq
regfile
iCache
dCache
unified
int alu
float alu
res bus
clock
Average power
0
5
10
15
8
16
32
64
128
Cache size (KB)

vpr



600
500
400
300
200
100
0
Useful cache entries
0
0.5
1
1.5
2
Executed instructions
x 10^7