Laboratory 10
Cache Memory and Processes

Computer Science 240

Caches = small, fast memories which contain current and recently/likely to be used data from the large, slower, main memory

Temporal and spatial locality make this possible

When programs have poor locality or characteristics that cause frequent cache misses, very poor (slow) performance can occur

The performance impact in such cases can be used to measure the dimensions of the cache

Cache Experiments on Real Hardware

- reps may be any positive number
- stride must be a power of 2 no larger than SIZE
- both functions perform the same task, but using different order, which affects the performance/time to complete the task

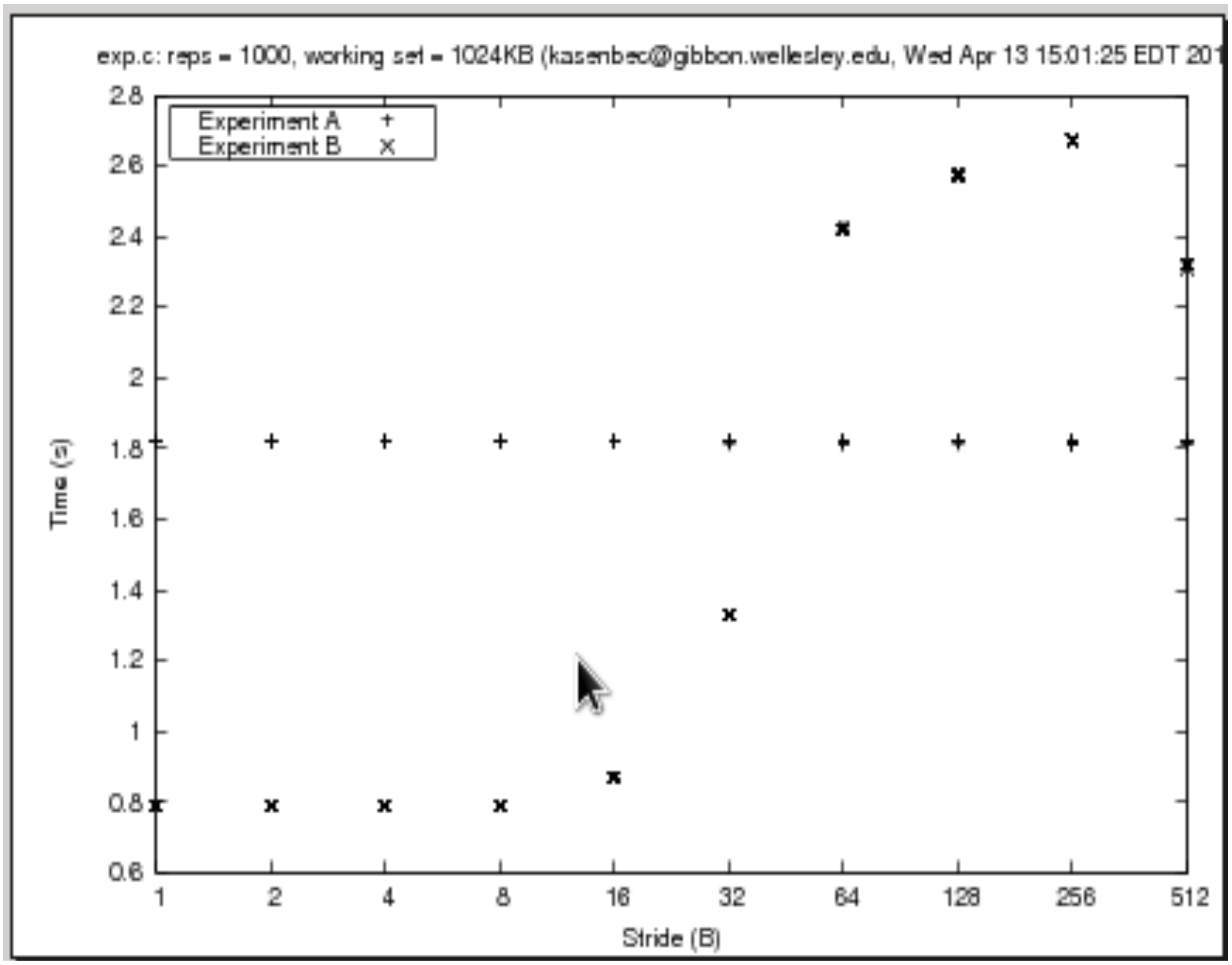
```
int experimentA(const int reps, const int stride) {

  assert(stride <= SIZE);

  int result = 0;

  for (int i = 0; i < SIZE; i += stride) {

    for (int j = 0; j < stride; j++) {

      for (int r = 0; r < reps; r++) {

        result += array[i+j];
        array[i+j]++;
      }
    }
  }

  return result;

}
```

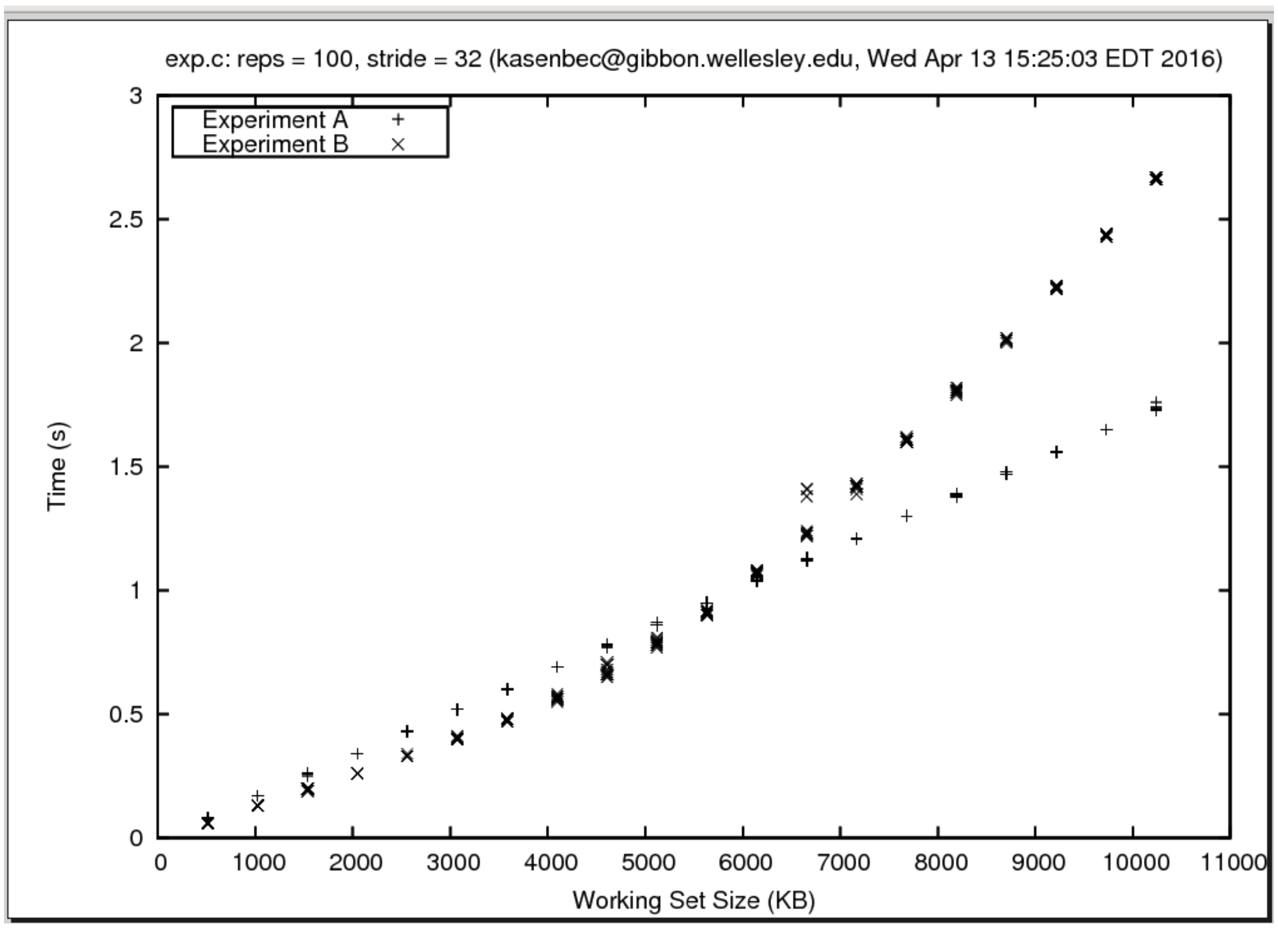
```
int experimentB(const int reps, const int stride) {

  assert(stride <= SIZE);

  int result = 0;

  for (int r = 0; r < reps; r++) {

    for (int j = 0; j < stride; j++) {

      for (int i = 0; i < SIZE; i += stride) {

        result += array[i+j];
        array[i+j]++;
      }
    }
  }

  return result;

}
```

Vary either size of array (working set size) or size of element (stride), and predict results: how do size of array and size of element (stride) affect the performance of either program?

Next use tools that allow you to run multiple trials, store the time for each experiment, and plot results.

The shape of the graphs helps you understand cache block size and cache size.




exp.c: reps = 100, stride = 32 (kasenbec@gibbon.wellesley.edu, Wed Apr 13 15:25:03 EDT 2016)
Experiment A +
Experiment B ×
Time (s)
Working Set Size (KB)
0
0.5
1
1.5
2
2.5
3
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
11000

Cache Sleuth

*U*se a cache **simulator** and attempt to automatically discover the dimensions of mystery caches based only on observations of hits and misses for a stream of memory accesses.

getLineSize(...) determines the line size of the cache.

getCapacity(...) determines the capacity of the cache.

getAssociatvity(...) determines the associativity of the cache.

Assumptions:

All caches use an LRU (least recently used) replacement policy.

All caches start empty (cold).