High Performance Computing for Science and Engineering

Exercise 2: Cache Design, Shared Memory
(Submissions Feedback)

Ivica Kičić
Computational Science and Engineering Laboratory
Question 1

- Forgot to take modulo -> $a[k] = (\ldots) \% N$
  - Causes overflow / undefined behaviour.
- $k = a[k] \% N$
  - Slow, whole thing should be $\sim 5$ cycles, here only DIV is $\sim 25$!
  - You won’t see the drop in performance in 1d!
- Result = $10^{15}$ op/s
  - Compiler optimized away the whole loop. Use `volatile`!
- $N-2$ in Sattolo’s algorithm
  - Rerun with $N-1$ and grade your solution with new results!
- Instead of $k = a[k]$ doing sequential $k = a[i]$, and other variants.
- Misc: For easier comparison, put all three lines in the same plot.
Question 2

- For a `double buf[]`, expression `buf[k]` already means a jump in `k*sizeof(double)` bytes!
- Only one pass --> too noisy plots.

Hardware is complex and unpredictable. Benchmark to be sure!
Question 3

- Not using tail properly.
  - Not O.K.:
    ```
    int s = tail;
    tail++;  
    slot[tid] = s;
    tail = tail == MAX_T - 1 ? 0 : tail + 1;
    ```

- OK:
  ```
  int s = tail.fetch_add(1) % MAX_T;
  int s = std::atomic_fetch_add(&tail, 1) % MAX_T;
  int s = tail++ % MAX_T;
  int s = ((tail += 1) - 1) % MAX_T;
  slot[tid] = s;

  int s;
  #pragma omp atomic capture
  s = tail++;
  ```

- rand() is NOT thread-safe
  - rand_r or std generator (one per thread)
  - srand(time(NULL)) in the loop can give same results for multiple threads

- critical should not be used!
  - It is a lock itself. Implementation loses purpose.

- volatile flag!