HIGH PERFORMANCE COMPUTING for SCIENCE & ENGINEERING (HPCSE) I

HS 2021

EXERCISE 04: POST DISCUSSION

Daniel Waelchli (wadaniel@ethz.ch)

Computational **S**cience and **E**ngineering **Lab**
ETH Zürich

26.11.2021

Diffusion

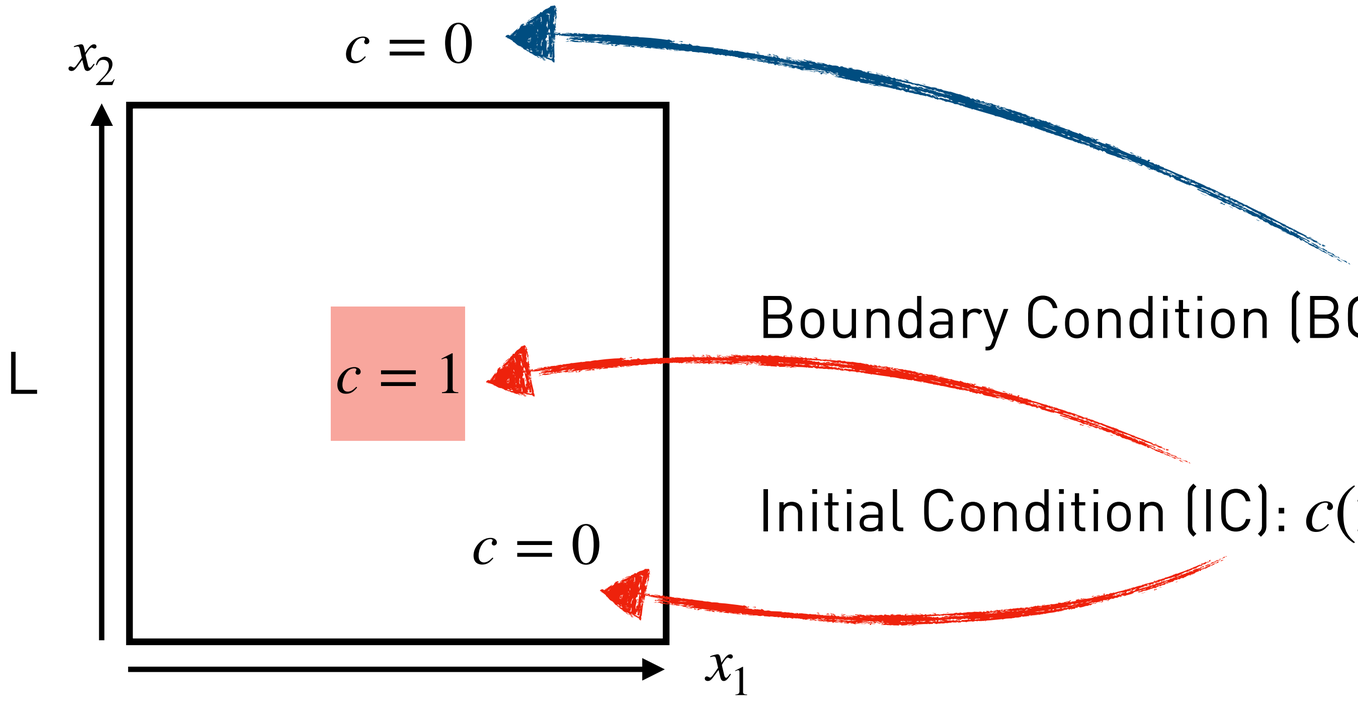
Diffusion is a physical process, which leads du a uniform distribution of particles.

In this exercise we consider the heat-flow in a 2D medium that can be described by:

$$\frac{\partial c(\mathbf{x},t)}{\partial t} = D\,\nabla^2 c(\mathbf{x},t)$$

$c(\mathbf{x},t)$ is the concentration of heat at position $\mathbf{x} = (x_1, x_2)$

D is a constant diffusion coefficient $[\frac{\partial x^2}{\partial t}]$



Boundary Condition (BC): $c(\mathbf{x},t) = 0$ for $t \geq 0$

Initial Condition (IC): $c(\mathbf{x},0) = \begin{cases} 1, & \text{if } ||\mathbf{x}||_1 \leq L/4 \\ 0, & \text{else} \end{cases}$

Question 1: Diffusion a

Question 1: Diffusion (25 points)

The diffusion of a substance can be described by the equation

$$\frac{\partial c(x,y,t)}{\partial t} = D\left(\frac{\partial^2 c(x,y,t)}{\partial x^2} + \frac{\partial^2 c(x,y,t)}{\partial y^2}\right), \tag{1}$$

where c is the concentration of the substance at position $(x,\ y)$ and at time t, and D is the diffusion constant. The diffusion process happens in the domain $|x| < L/2$ and $|y| < L/2$. The concentration is zero on the boundaries of the domain for $t \geq 0$. The initial concentration is

$$c(x,y,0) = \begin{cases} 1, & \text{if } |x| < L/4 \text{ and } |y| < L/4, \\ 0, & \text{otherwise.} \end{cases}$$

a) Write down the 2-dimensional discretized diffusion process for eq. (1). Assume a uniform grid with spacing h and a central finite difference scheme in space and forward Euler time integration. For the forward Euler integration assume time intervals of size dt. Make sure that you annotate all variables.

The corresponding discretized equation is:

$$c_{i,j}^{t+1} = c_{i,j}^{t} + D\frac{dt}{h^2}(c_{i+1,j}^{t} + c_{i-1,j}^{t} + c_{i,j+1}^{t} + c_{i,j-1}^{t} - 4c_{i,j}^{t}) \tag{2}$$

where $c_{i,j}^{t} = c(i*h, j*h, t)$ is the concentration at time t for $x = i*h$ and $y = j*h$.

Grading scheme:

- 5p: Correct final equation
- -1p: If annotations missing (e.g. $c_{i,j}^{t} = c(i*h, j*h, t)$)

Total: 5p

Question 1: Diffusion b

$$\rho^{t+1}e^{ik(x_i+x_j)} = \rho^t e^{ik(x_i+x_j)} + D\frac{dt}{h^2}\rho^t(e^{ik(x_i+h+x_j)} + e^{ik(x_i-h+x_j)} + e^{ik(x_i+x_j+h)} + e^{ik(x_i+x_j-h)} - 4e^{ik(x_i+x_j)})$$
$$= \rho^t e^{ik(x_i+x_j)} + D\frac{dt}{h^2}\rho^t(2e^{ik(x_i+h+x_j)} + 2e^{ik(x_i-h+x_j)} - 4e^{ik(x_i+x_j)}) .$$

2p

Dividing both sides by $\rho^t e^{ik(x_i+x_j)}$ gives

$$\rho = 1 + \frac{Ddt}{h^2}(4\cos(kh) - 4) .$$ 1p

For stability we must have $|\rho| \leq 1$. This yields the upper bound

$$\rho_{max} = \frac{h^2}{4D}$$ 1p

The implementation can be found in `solution_code/q1/diffusion.cpp`. 1p

Grading scheme:
Inside advance:

- 2p: Correct initial assumption and replacement.
- 1p: Solving for ρ.
- 1p: Finding ρ_{max}.
- 1p: Implementation.

Total: 5p

Question 1: Diffusion c

c) Based on the discretization found in the first subquestion, provide a cache-friendly implementation of the diffusion equation in the method advance. I.e. avoid copying of memory if possible and mind the access patterns of the memory. Blocking must not be implemented.

```
  void advance() {
    /* Central differences in space, forward Euler in time, Dirichlet BCs */
    // TODO: 1c Implement central difference in space, forward Euler in time
    // TODO: 1e Parallelize diffusion with OpenMP
#pragma omp parallel for collapse(2)
    for (size_t i = 1; i <= N; ++i)
      for (size_t j = 1; j <= N; ++j)
        c_tmp[i * (N + 2) + j] =
            c[i * (N + 2) + j] +
            aux * (c[i * (N + 2) + (j + 1)] + c[i * (N + 2) + (j - 1)] +
                   c[(i + 1) * (N + 2) + j] + c[(i - 1) * (N + 2) + j] -
                   4 * c[i * (N + 2) + j]);
    std::swap(c_tmp, c);
  }
```

start at index i,j = 1 because of BC

efficient memory access pattern 1p

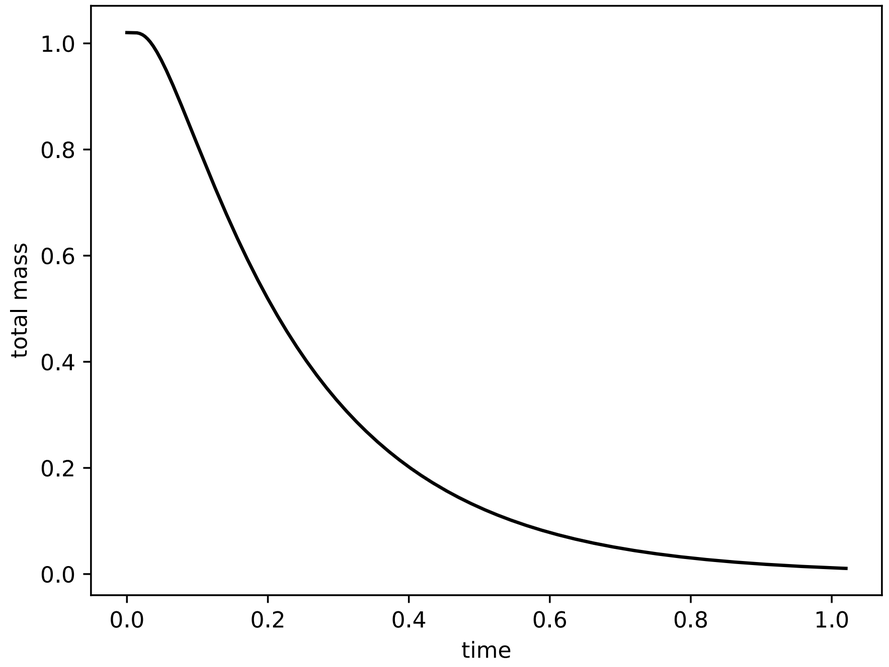
dont assign/copy c_tmp into c, just exchange pointers 1p

correctness 3p

Question 1: Diffusion d

d) Plot the total concentration as a function of time for $t \in [0, 0.5]$ using $D = 1$, $L = 2$ and $N = 100$. The concentration can be read from the file `diagnostics.dat` (column 0 and 1). Qualitatively explain the behaviour of the graph in less than 3 sentences, is this result expected?

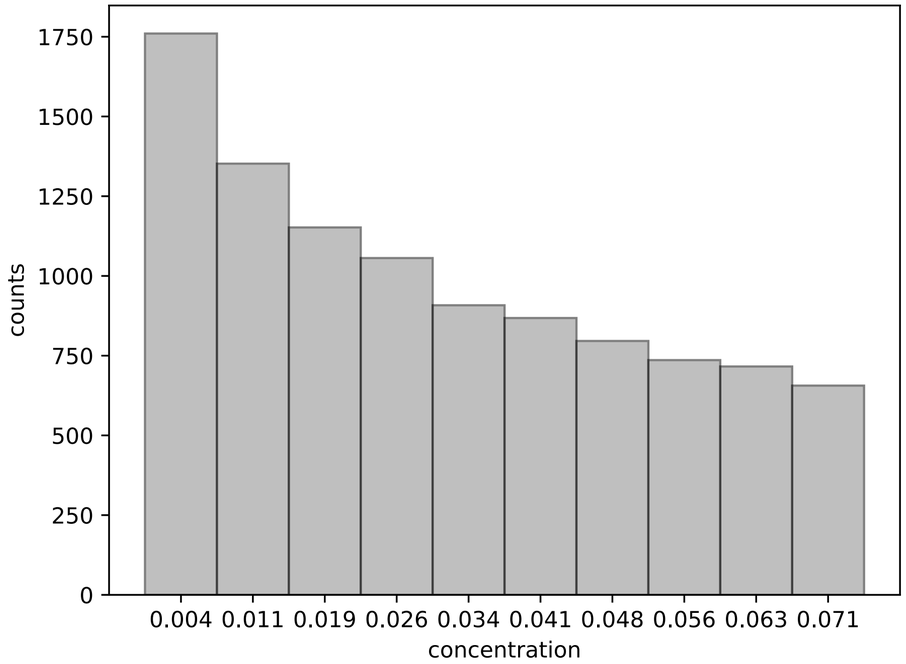
Total Concentration



total concentraction decreases 2p

argumentation 1p

Bins (Optional)



if labels missing, value range not clear -1p

Question 1: Diffusion e

e) Parallelize the diffusion process (your implementation from subquestion 1c) in the method advance using OpenMP.

f) Parallelize the integration of the concentration (marked with TODO in compute_diagnostics) and the calculation of the histogram (marked with TODO in the method compute_histogram) using OpenMP.

```
  void advance() {
    /* Central differences in space, forward Euler in time, Dirichlet BCs */
    // TODO: 1c Implement central difference in space, forward Euler in time
    // TODO: 1e Parallelize diffusion with OpenMP
#pragma omp parallel for collapse(2)
    for (size_t i = 1; i <= N; ++i)
      for (size_t j = 1; j <= N; ++j)
        c_tmp[i * (N + 2) + j] =
            c[i * (N + 2) + j] +
            aux * (c[i * (N + 2) + (j + 1)] + c[i * (N + 2) + (j - 1)] +
                   c[(i + 1) * (N + 2) + j] + c[(i - 1) * (N + 2) + j] -
                   4 * c[i * (N + 2) + j]);
    std::swap(c_tmp, c);
  }
```

parallel for loop 2p

(collapse optional)

```
    /* Integration to compute total concentration */
    // TODO: 1f Parallelize integration with OpenMP
#pragma omp parallel for collapse(2) reduction(+ : amount)
    for (size_t i = 1; i <= N; ++i)
      for (size_t j = 1; j <= N; ++j)
        amount += c[i * (N + 2) + j];

    amount *= (h * h);

    printf("t = %lf amount = %lf\n", t, amount);
    return amount;
  }
```

parallel for loop

reduction of var amount 1p

(collapse optional)

Question 1: Diffusion f (cont.)

```
  void compute_histogram(std::vector<int> &hist) {
    /* number of bins */
    const size_t M = hist.size();

    /* Initialize max and min concentration */
    double max_c, min_c;
    max_c = c[1 * (N + 2) + 1];
    min_c = c[1 * (N + 2) + 1];

    /* Find max and min concentration values */
    // TODO: 1f Parallelize max_c and min_c initialization
#pragma omp parallel for reduction(min : min_c) reduction(max : max_c)
    for (size_t i = 1; i <= N; ++i)
      for (size_t j = 1; j <= N; ++j) {
        double c0 = c[i * (N + 2) + j];
        if (c0 > max_c)
          max_c = c0;
        if (c0 < min_c)
          min_c = c0;
      }

    const double epsilon = 1e-8;
    double dc = (max_c - min_c + epsilon) / M;

    /* Accumulate equispaced bins */
    // TODO: 1f Parallelize bin accumulation
#pragma omp parallel
    {
      // local calculation of bins
      std::vector<int> local_hist(M, 0);
#pragma omp for nowait // nowait optional
      for (size_t i = 1; i <= N; ++i) {
        for (size_t j = 1; j <= N; ++j) {
          size_t bin = (c[i * (N + 2) + j] - min_c) / dc;
          local_hist[bin]++;
        }
      }
      // global aggregation of local histograms
#pragma omp critical
      for (size_t i = 0; i < M; ++i)
        hist[i] += local_hist[i];
    }
```

parallel for loop
reduction of vars min_c, max_c 2p
(collapse optional, not recommended)
move declaration c0 inside or mark it private

“ideal” solution:
create local hist and aggregate later 2p

“naive” solution
(1/2 points, because slow

(1p)

```
    /* ALTERNATIVE NAIVE SOLUTION (1/2pts, very slow)
     *
     * Accumulate equispaced bins
    // TODO: 1f Parallelize bin accumulation
#pragma omp parallel
    for (size_t i = 1; i <= N; ++i) {
      for (size_t j = 1; j <= N; ++j) {
        size_t bin = (c[i * (N + 2) + j] - min_c) / dc;
#pragma omp atomic
          hist[bin]++;
        }
      }
    */
  }
```

Question 2: PSE b

Question 2: Particle Strength Exchange (20 points)

Consider the diffusion equation eq. (1) from the previous exercise. We want to utilize the particle strength exchange (PSE) method to solve this diffusion problem. Instead of discretizing the field $c(x, y, t)$ on a grid, we will use a collection of N particles. A particle represents a small "volume" of the field and is defined by its position $\mathbf{x}_i$ and field value $\phi_i = \pi_i(t)$. In this exercise, we assume the volume of each particle is equal $V_i = V_{total}/N = L^2/N$. We rewrite eq. (1) as a system of equations on particles:

$$\frac{\partial \phi_i}{\partial t} = \frac{D}{\epsilon^2} \sum_{j=1}^{N} V_j (\phi_j - \phi_i) \eta_\epsilon (x_j - x_i) \,, \tag{2}$$

where $\eta_\epsilon(r)$ is a kernel representing the Laplacian operator, and ϵ a scale constant. In this exercise we consider the following kernel:

$$\eta_\epsilon(\mathbf{r}) = \frac{4}{\epsilon^2 \pi} e^{-\frac{1}{\epsilon^2}|r|^2}. \tag{3}$$

Assume the same initial and boundary conditions as in the previous exercise.

In the following subquestions, you will work with the codes provided in `/ex04/skeleton_code/q2/`. Please have a look at the README file for further information. We recommend compiling and running the code on the Euler cluster.

a) You are given a skeleton code that initializes the particle positions and values. Get familiar with the skeleton code. Use `make run` and `make plot` to run the code and the the scripts.

b) Extend the provided skeleton code to compute the right-hand side of eq. (2) using eq. (3) for the kernel η_ϵ. Reuse pair-wise quantities and reduce the number of operations. Implement the forward Euler scheme for the integration of the eq. (2).

Total 5p

N^2 iterations -1p

Particles on bounds wrong -1p

d) Parallelize the `timestep` method using OpenMP. For this, you need to modify the `Makefile` and include the library in your source code.

```
/*
 * Perform a single timestep. Compute RHS of Eq. (2) and update values of phi_i.
 */
// TODO  2d: Parallelize this function with OpenMP.
void timestep(void) {
  const double volume = DOMAIN_SIZE * DOMAIN_SIZE / N;
  const double factor = D * volume / (eps * eps);

  // TODO 2b: Implement the RHS.
#pragma omp parallel for
  for (size_t i = 0; i < N; ++i) {
    double sum = 0;
    for (size_t j = i + 1; j < N; ++j) {
      const double tmp =
          (phi[j] - phi[i]) * gaussian_eps(x[i] - x[j], y[i] - y[j]);
      sum += tmp;
#pragma omp atomic
      dphi[j] -= factor * tmp;
    }
    dphi[i] += factor * sum;
  }

  // TODO 2b: Implement the forward Euler update of phi_i.
#pragma omp parallel for collapse(2)
  for (size_t i = 1; i < SQRT_N - 1;
       ++i) // skip updating of particles on x boundaries
    for (size_t j = 1; j < SQRT_N - 1;
         ++j) // skip updating of particles on y boundaries
    {
      const size_t k = i * SQRT_N + j;
      phi[k] += dt * dphi[k];
      dphi[k] = 0.; // reset dphi
    }
}
```

Question 2: PSE c

c) Considering the domain size and the number of particles, what is qualitatively the distance h between neighboring particles? Parameter ϵ determines the spread of the kernel. Run the code for different values of ϵ. Experiment with

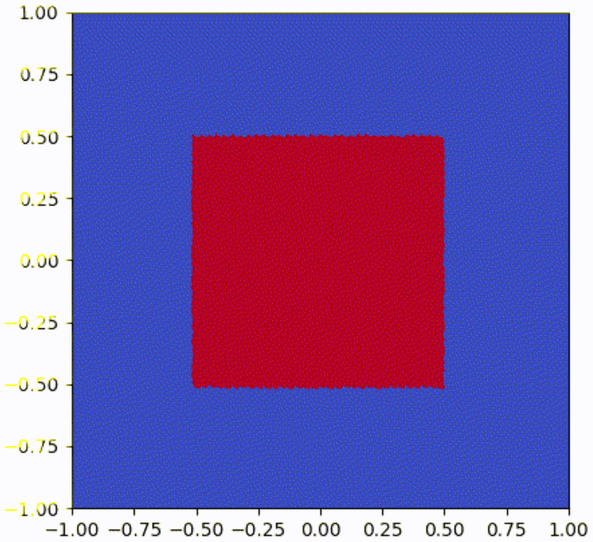
- $\epsilon << h$,
- $\epsilon \approx h$,
- and $\epsilon >> h$.

What do you observe for different cases? You can visualize the output of you runs with `make plot`.

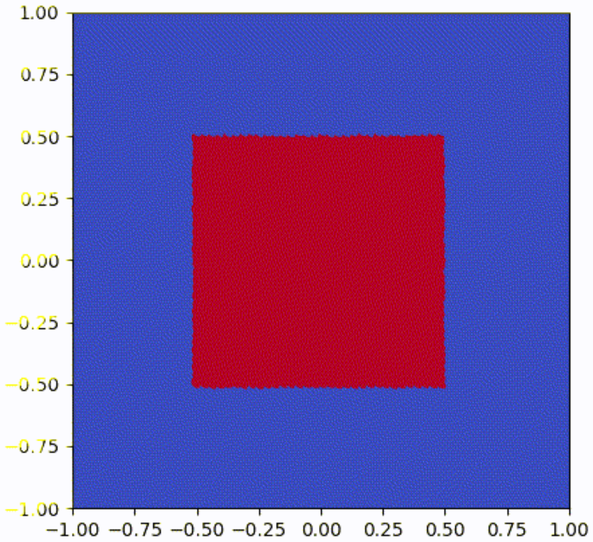
argumentation 3p

N particles $\rightarrow$ $\sqrt{N}$ particles per dim $\quad h \approx \frac{L}{\sqrt{N}} = 0.03125$ 2p (Input to program is $\sqrt{N} = 64$)

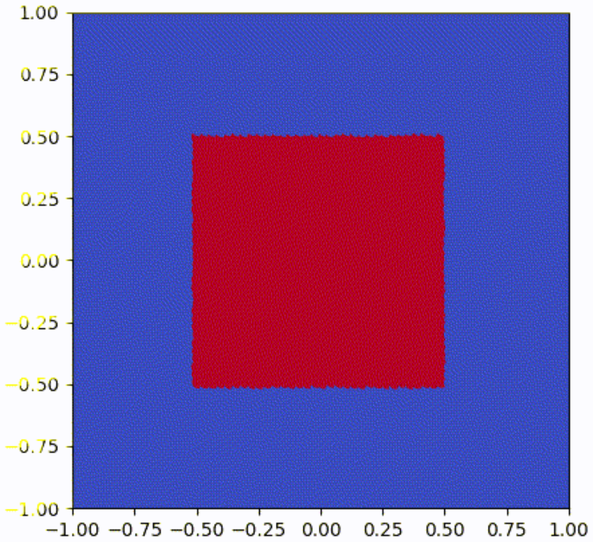
$\epsilon = 0.01$



$\epsilon = 0.03$

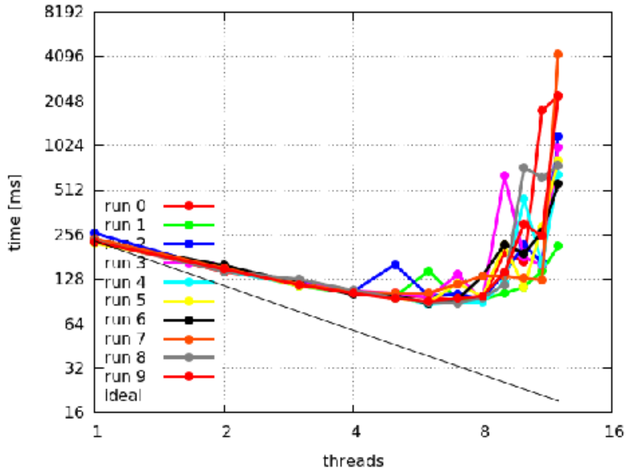


$\epsilon = 0.3$

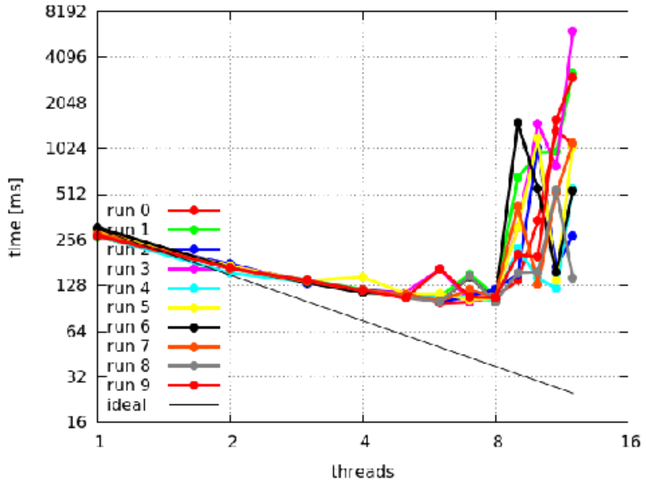


Question 2: Benchmarking (Q1)

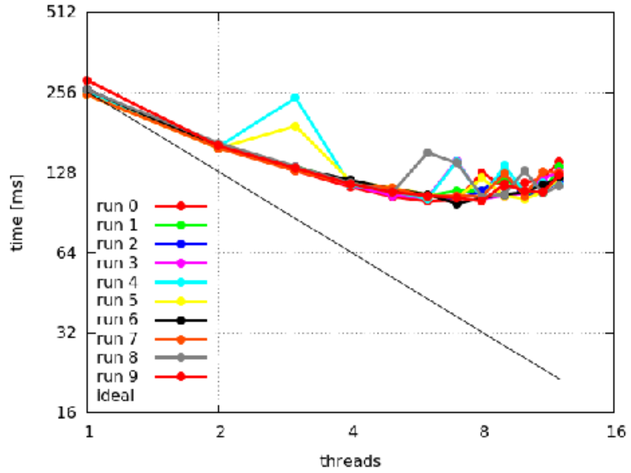
Version: No Collapse



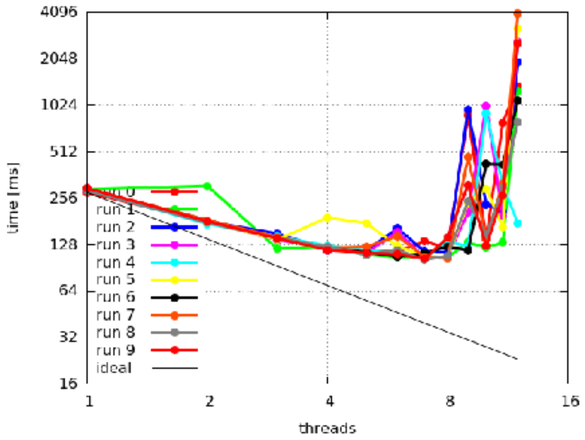
Version: 1x Collapse



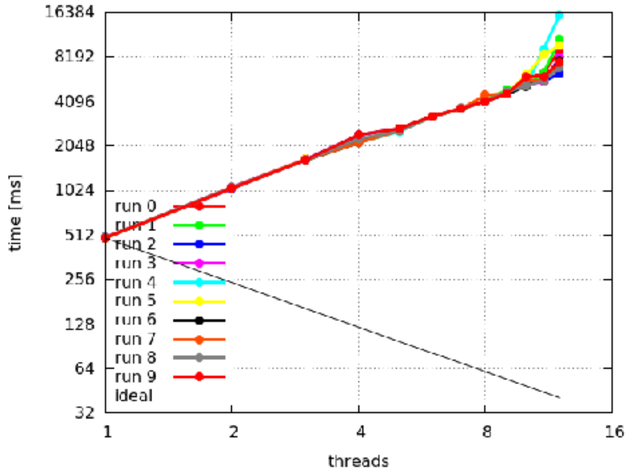
Version: 2x Collapse



Version: 3x Collapse



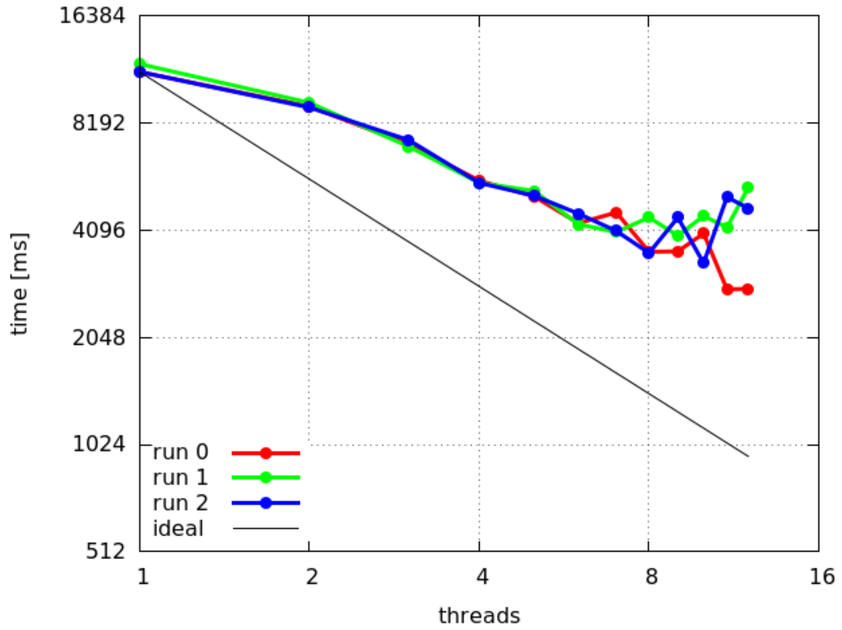
Version: Atomic Binning



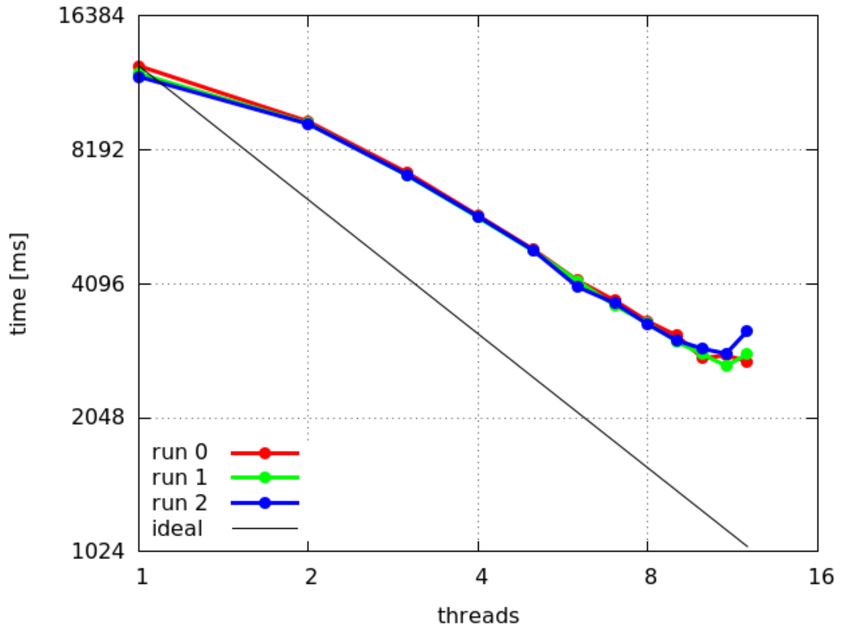
Measurements done on Euler interactive shell (-n12)

Question 2: Benchmarking (Q2)

Version: No Collapse



Version: 1x Collapse



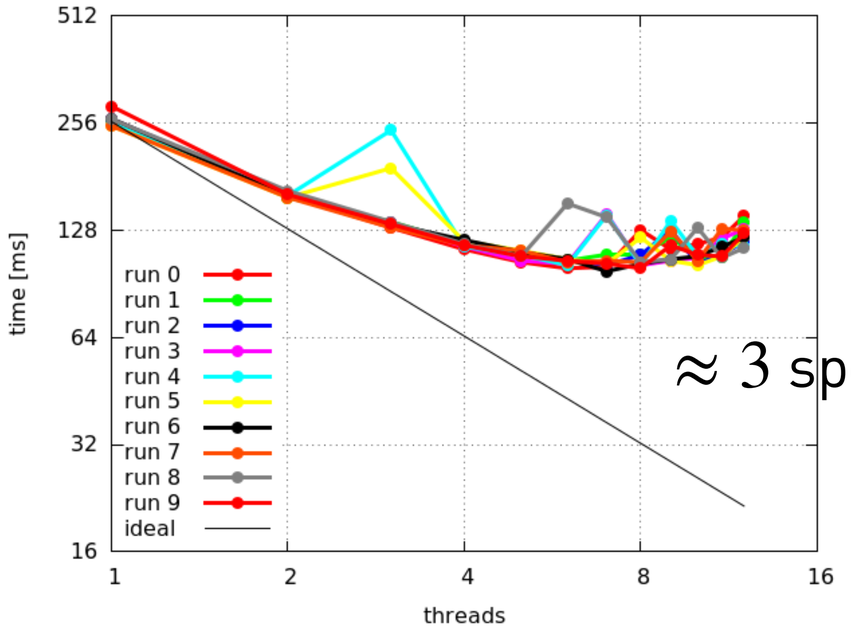
Measurements done on Euler interactive shell (-n12)

Question 2: Comparison e

Strong Scaling 1p

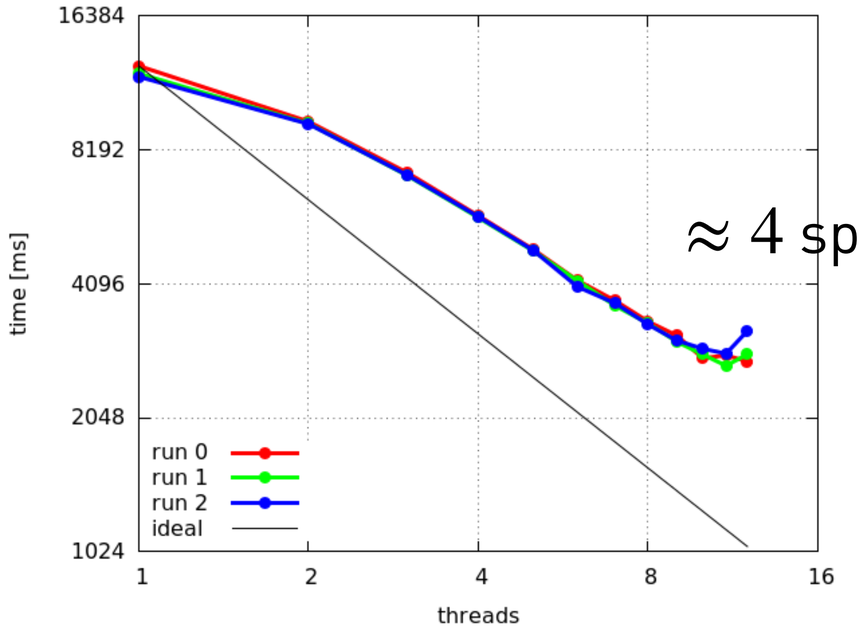
Plots 2p

Finite Difference: 2x Collapse



≈ 3 speed-up

PSE: 1x Collapse



≈ 4 speed-up

Observation & justification 2p

Measurements done on Euler interactive shell (-n12)

Comments & Questions

Thank you.