PTfS-CAM

Project: Modelling 2D steady-state heat equation

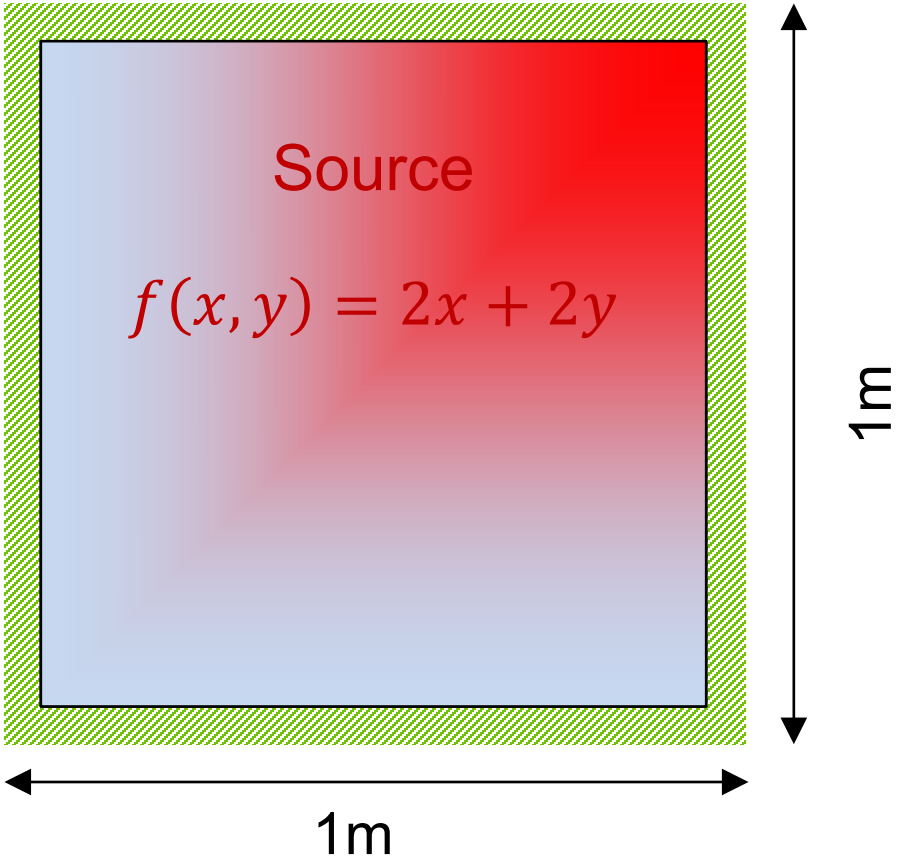
Part 2

Warning:
This session will be recorded. If you want to ask a question and do not want it to appear on the recording, wait till the end. You may also change your display name in Zoom.

Scenario: Heat dissipation on a rectangular plate

Find steady state temperature distribution inside the plate!

Boundary
$T(\varphi) = 0$



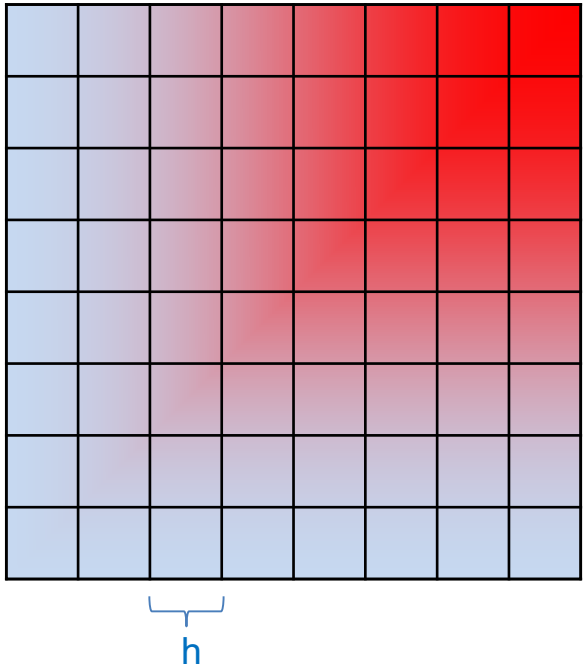
Discretization

$$-\Delta u = -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f$$

Use Finite Difference Method (FDM) for discretization

$$\Rightarrow -\Delta u = -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)(x,y) \approx$$

$$\frac{1}{h^2}\left(4u(x,y) - u(x-h,y) - u(x+h,y) - u(x,y-h) - u(x,y+h)\right)$$



Solving linear system

Solve for $u : -\Delta u = f$

1. Use Conjugate Gradient (CG)

2. Use Preconditioned Conjugate Gradient (PCG) with symmetric Gauss-Seidel preconditioning

Your tasks

1. Copy the code from ~unrz002h/ptfs_cam_2021/project :
   `cp -r ~unrz002h/ptfs_cam_2022/project <target destination>`

2. Build the code using the given Makefile, i.e., just type **`CXX=icpx make`**

3. To switch on LIKWID measurement set the LIKWID flag to ‘on’, i.e.,
   `LIKWID=on CXX=icpx make`

4. Check for code correctness using the **test** executable: **`./test`**

5. To run the actual code us the **perf** executable:
   `./perf num_grids_y num_grids_x`

6. If all tests pass, **parallelize building blocks using OpenMP**. Always observe correctness!

7. Are there any possible optimizations that you could do in the CG and PCG solver implemented in **`SolverClass::(P)CG`** (Solver.cpp)? If so, implement them!

Your tasks (new)

8. Calculate roofline predictions in [LUP/s] for CG and PCG solvers on 1 ccNUMA domain (18 cores) of Fritz. Calculate for two grid sizes : 2000×20000 and 1000×400000. The last dimension is in x-direction (innermost). Does the performance change? Why?

9. Check whether you attain the roofline performance by running the code on 1 ccNUMA domain of Fritz for the two dimensions given above. Timings are already included. Run the code using following command.
   `./perf  num_grids_y  num_grids_x`

10. Measure the code balance in [bytes/LUP] of 'applyStencil' and 'GSPreCon' kernels on 1 ccNUMA domain (18 threads) for both the grid dimension and comment whether it agrees with your model. You can use LIKWID for the measurement.
    - To switch on LIKWID measurement set the LIKWID flag to 'on', i.e.,
      `LIKWID=on CXX=icpc make`

11. How does the CG and PCG code scale from 1 ccNUMA domain to 1 node (4 ccNUMA domains, 72 cores) of Fritz? Does it scale perfectly? If not why? Can you fix it?

Performance prediction for entire algorithm

PCG example

```
while( (iter<niter) && (res_norm > tol*tol) )
        v   = A p
        λ   = α0 / <v,p>
        x   = x + λ p
        r   = r - λ v
        z   = Pr
        α1  = <r,z>
        p   = z + (α1/α0) p
        α0  = α1
        ++ iter
```

Multiple kernels

Performance prediction for entire algorithm : example

```
void scale(int n,…)
{
    for(int i=0; i<n; ++i)
        a[i] = c*b[i];
}

void add(int n,…)
{
    for(int i=0; i<n; ++i)
        a[i] = b[i] + d*c[i];
}

int main()
{
    …
    scale(1e8,…);
    add(1e8,…);
        …
    return 0;
}
```

$I_{scale} = \frac{1}{16}\frac{Flop}{Byte}$, $P^{max}_{scale} = 1 * 8 * 2 * 18 = 288\ GFlop/s$

ST SIMD f #cores

$P_{scale} = \min(288, 76/16) = 4.75\ GFlop/s$

$I_{add} = \frac{2}{24}\frac{Flop}{Byte}$, $P^{max}_{add} = 2 * 8 * 2 * 18 = 576\ GFlop/s$

$P_{add} = \min(576, 76*2/24) = 6.3\ GFlop/s$

Is it $P_{total} = P_{scale} + P_{add}$? **NO**

But $T_{total} = T_{scale} + T_{add}$

$T_{total} = \frac{1e8*1}{P_{scale}} + \frac{1e8*2}{P_{add}} = 0.0528$ s

=> $P_{total} = \frac{1e8}{T_{total}} = 1.9\ GIT/s$

Things to take care

- Think to use loop fusion wherever necessary.
- For debugging please compile code as: **`CXX=icpx make EXTRA_FLAGS=-DDEBUG`**
- Sometimes it's useful for debugging to visualize your arrays. Use the function **`writeGnuplotFile`** and plot using **`splot`** in gnuplot if needed.
- Take particular care with parallelizing the Gauss-Seidel preconditioner.
- Use Fritz (Ice Lake) for getting your performance results.
- Fix clock frequency to 2.0 GHz
- Check if the measurements are reproducible. (Think of pinning, scheduling, and clock frequency).
- Request a dedicated node for measuring performance.

Things to take care

- Remember pinning (`-C` with **`likwid-perfctr`**) when doing performance counter measurements. Use `-m` for using markers, instead of end-to-end measurement.
- When measuring parallel loops with **`likwid-perfctr`**, LIKWID_MARKER_START and LIKWID_MARKER_STOP should be called by all threads.
- Remember **`likwid-perfctr`** can incur some overheads, so for performance measurement better run without **`likwid-perfctr`**.
- Remember to arrive at final roofline model of (P)CG, you would need to stitch performance models of different kernel. It might be convenient to use time based model (see previous slide). If you wish use Excel sheet for this.
- For roofline modeling you can assume the memory bandwidth of 1 ccNUMA domain of Fritz is 76 GB/s.
- Remember Fritz has a write-allocate avoiding mechanism that can kick in some cases.

Submission

- Submit the code after doing tasks until Sept 2, 2022 at PTfS Moodle (link will be available). Please compress and submit only a single (ZIP) file. Include code and the report in submission.
- The report should contain all the details necessary to reproduce your measurements.
- All submitted code should be compilable by just typing **`make`** (Makefile is already provided),
- Your code should pass all the tests.
- Both executables (**`test`** and **`perf`**) should run without segmentation faults.
- While submitting report expected roofline performance and the measured performance, use $\left[\frac{LUP}{s}\right]$ (= $\left[\frac{IT}{s}\right]$) as performance metric, see definition in **`perf.cpp`**.
- If there is any substantial deviation between these values, please explain plausible cause if any.

Final remark

- In the exam you will be definitely asked questions based on this exercise.
- Happy Coding !!!