Modelling 2D steady-state heat equation

PTfS-CAM Project

Part II

PTfS 2020

Solving the linear system

Solve for u : $A\,u = f$

1. Use Conjugate Gradient (CG)

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

Your TASKS – part II

1. Calculate roofline predictions in [LUP/s] for CG and PCG solvers on 1 socket (10 cores) of Emmy. Calculate for two grid sizes : 1000×20000 and 1000×100000. Does the performance change? Why?

2. Check whether you attain the roofline performance by running the code on 1 socket of Emmy. Timings are already included. Run the code using following command.
   `./perf  num_grids_y  num_grids_x`

3. Measure the code balance in [bytes/LUP] of 'APPLY_STENCIL' kernel on 1 socket (10 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`**

4. How does the code scale from 1 socket to 1 node (2 sockets, 20 cores) of Emmy? Does it scale perfectly? If not Why?

Chained roofline model

PCG example

```
r = b – A x
res_norm = <r,r>
z = P r0
α0 = <r,z>
p = z

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

Chained roofline model – a simple 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;
}
```

$I_{scale} = \frac{1}{24}\frac{Flop}{Byte}$, $P_{scale}^{max} = \frac{1}{2} * 4 * 2.2 * 10 = 44\ GFlop/s$

ST, SIMD, f, CORES

$P_{scale} = \min(44, 40/24) = 1.67\ GFlop/s$

$I_{add} = \frac{2}{32}\frac{Flop}{Byte}$, $P_{add}^{max} = \frac{1}{2} * 2 * 4 * 2.2 * 10 = 88\ GFlop/s$

$P_{add} = \min(88, 40*2/32) = 2.5\ GFlop/s$

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

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

$T_{total} = \frac{1e8*1}{P_{scale}} + \frac{1e8*2}{P_{add}} = 0.1398$ s

$\Rightarrow P_{total} = \frac{1e8}{T_{total}} = 0.7\ GIT/s$

Things to take care of

- Use EMMY (Ivy Bridge) for getting your performance results.
- Check if the measurements are reproducible. (Think of pinning, scheduling, and clock frequency).
- 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 run without 'LIKWID=on'.
- 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).

Submission

- Submit the code after doing tasks until July 27, 2020 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 segfaults.
- While submitting report expected roofline performance and the measured performance, use $\left\lceil \frac{LUP}{s} \right\rceil$ (= $\left\lceil \frac{IT}{s} \right\rceil$) 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 !!!