NHR FAU

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Hands on: Part 1

likwid-topology, likwid-pin, SpMV CRS code

Setting up

- Allocate an exclusive interactive node on OOKAMI
  - srun --nodes=1 --ntasks-per-node=1 --cpus-per-task=48 --exclusive -t 2:00:00 -p short --pty /bin/bash –l
- git clone https://github.com/RRZE-HPC/A64FX_SpMV_hands-on
- cd A64FX_SpMV_hands-on
- module load gcc/11.1.0
- source /lustre/projects/global/samples/FAU_webinar/likwid-ur/sourceme.sh
- make

Run SpMV code

- `OMP_SCHEDULE=static OMP_NUM_THREADS=48 ./spmv-SELLC-GCC`
- Try repeating the same command
- Observe the fluctuation in performance

Run the code with pinning

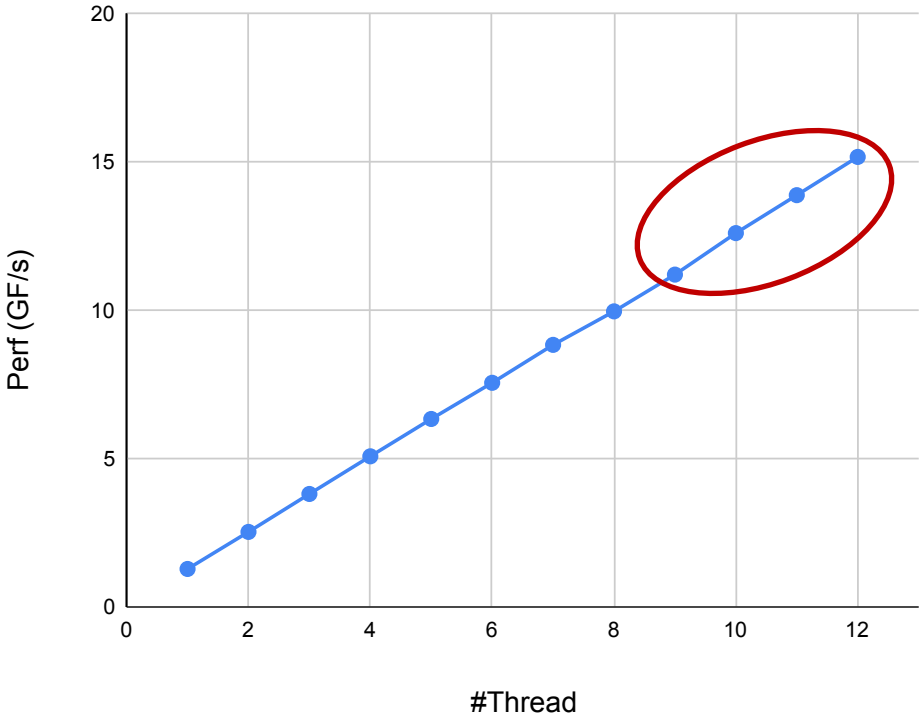
- likwid-topology → shows the topology
- likwid-pin –h → shows help
- likwid-pin –p → prints available affinity domains
- OMP_SCHEDULE=static OMP_NUM_THREADS=48 likwid-pin -c 0-47 ./spmv-SELLC-GCC
- Repeat the command → observe the improved performance without much fluctuation.

Let’s do some performance analysis with CRS SpMV

- Explore the CRS code a bit
- Lets see scaling of the code (spmv-CRS-GCC) within a socket, using likwid-pin.
- `OMP_SCHEDULE=static OMP_NUM_THREADS=${thread} likwid-pin -c 0-$((thread-1)) ./spmv-CRS-GCC`
- Run the script `./scaling_crs.sh`

Let’s do some performance analysis with CRS SpMV

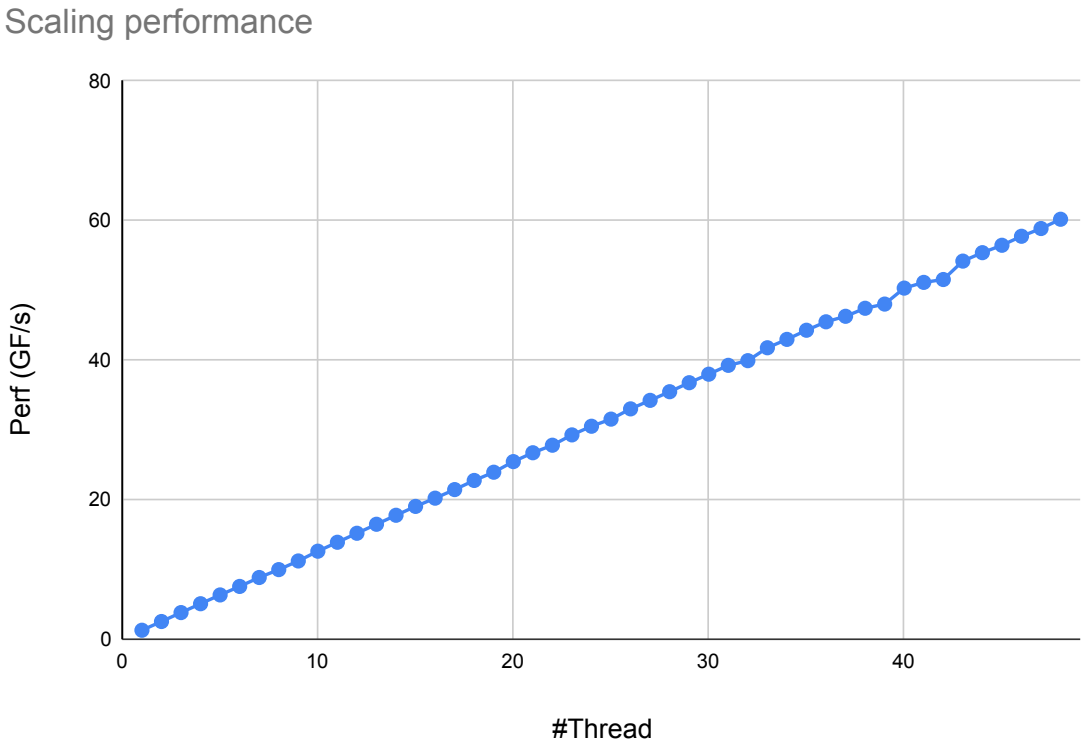
Scaling performance



Where is the saturation ?

1 socket, HPCG-128^3 matrix

Let’s do some performance analysis with CRS SpMV



1 full node, HPCG-128^3 matrix

Need a deep dive to understand the problem

- Generate the assembly code
  - make asm
- Coming up: analyze the code using OSACA