Parallel Computing Hackathon

Alexander Ayriyan, Ján Buša Jr.

Meshcheryakov Laboratory of Information Technologies, JINR, Dubna, Russia

Dubna
18. 10. 2023

Govorun + HybriLIT (`hlit.jinr.ru`)

- Supercomputer Govorun comprising of 3 different node types
  1. Pure CPU node – 88 nodes, 2× Intel Xeon Platinum 8280 (28c/56t), 192 GB RAM – `cascade`
  2. Xeon Phi node – 21 nodes, Intel Xeon Phi 7190 (72c/288t), 96 GB RAM – `knl`
  3. GPU node – 8 nodes, 4 × (8× NVIDIA Tesla V100 16GB), 4 × (8× NVIDIA Tesla A100 80GB)
- Platform HybriLIT
  - Pure CPU nodes – mainly based on Intel Xeon E5-2695 v2 (12c/24t) – `cpu`
  - GPU nodes – several (2 or 3) GPUs NVIDIA Tesla K20, K40, K80 – `gpu`
- Shared filesystem (Lustre, zfs, DAOS – 8PB)
- SLURM workload manager

Few Preliminary Steps

- to connect to HybriLIT from Linux/MacOS use terminal (find it) and from Windows use putty (download it)
- your login/password is

  user: `tut[001-100]`

  password: `itschool23`
- connect to `hydra.jinr.ru` using either terminal or putty

Basics of Terminal Environment

Moving around:

- `cd dirname` — enter directory `dirname`
- `cd ..` — go one directory up
- `cd ../dirname` — go one directory up and into `dirname`
- `ls` — show (list) all non-hidden files in current directory
- `pwd` — show path to current working directory
- `[tab]` (keyboard key) — fill in file name
- `cat | more | less [filename]` — show contents of `[filename]`
- `Arrow up/down` — repeat previous commands

Edit files: `nano` [filename]
Twin panel manager: `mc` (also good for editing files)

Execution of Programs

Run: `cd && mkdir tmp && cd tmp/`
To run some program:

- `tutorial.sh` — file describing requested resources and commands to be executed (`nano tutorial.sh`):

```
#!/bin/bash
#SBATCH -p tut
# SBATCH -ntasks=1
# SBATCH -cpus-per-task=4
time sleep 60s
echo 'Hello'
```

- `sbatch tutorial.sh` — start execution of the program set inside `tutorial.sh`. After termination you will get file `slurm-XXXXXX.out` containing result of calculations.
- `squeue` – see queue (status of jobs)

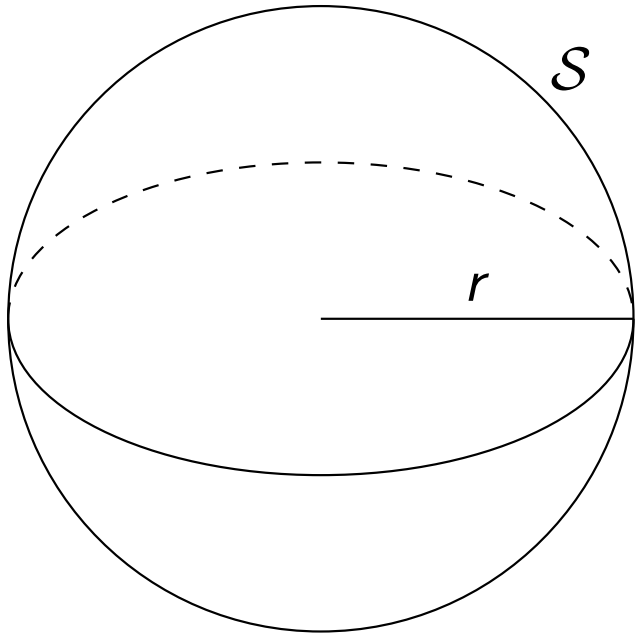
Volume of a Sphere

A volume V of a sphere $\mathcal{S}$ with the radius r is:

$$V = \frac{4}{3}\pi r^3$$

Knowing the volume and radius, it is easy to estimate value of π as:

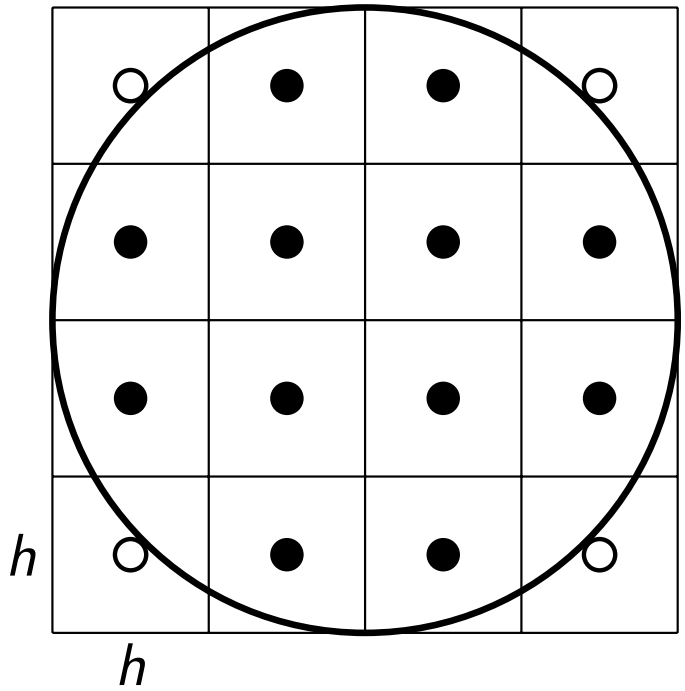
$$\pi = \frac{3}{4}\frac{V}{r^3}$$



Discretization of the Space Around the Sphere (2D View)

We fit sphere into a cube, discretize the area bounded by the cube – split it into many small blocks with centers:

$$\begin{aligned} x_i &= x_{\min} + h \cdot (i + 0.5) \\ y_j &= y_{\min} + h \cdot (j + 0.5) \\ z_k &= z_{\min} + h \cdot (k + 0.5) \end{aligned}$$



and assign value 1 to the blocks, whose center falls into the sphere (0 outside):

$$v(x_i, y_j, z_k) = \begin{cases} 1 & \text{if } \sqrt{x_i^2 + y_j^2 + z_k^2} \leq \mathrm{r} \\ 0 & \text{otherwise} \end{cases}$$

Estimate of π

Summing all elementary volumes and multiplying by h^3 we get the estimate for volume of the sphere

$$V \approx \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} [h^3 \cdot v(x_i, y_j, z_k)],$$

where n is an approximation factor. If we choose our discretization in such a way, that $r = 1$, we can estimate π as

$$\pi \approx \frac{3}{4} V.$$

First Program (serial)

```
double volume = 0;
double x, y, z;
double h = 2.0 / static_cast<double>(sizeN);
double elementaryVol = h * h * h;
for (unsigned int kdx = 0; kdx < sizeN; ++kdx){
  for (unsigned int jdx = 0; jdx < sizeN; ++jdx){
    for (unsigned int idx = 0; idx < sizeN; ++idx){
      z = -1.0 + h * (kdx + 0.5);
      y = -1.0 + h * (jdx + 0.5);
      x = -1.0 + h * (idx + 0.5);
      if (sqrt(x*x + y*y + z*z) <= 1.0)
        volume = volume + elementaryVol;
      }
    }
  }
double approxPi = volume * 3.0/4.0;
```

Execute Program

```
cd
wget t.ly/TOBN8 -O h.zip && unzip h.zip && rm h.zip
cd hackaton2023/cpp

make
sbatch pi_cpp.sh
squeue
cat slurm-XX.out

g++ -O0 -o pi_cpp pi.cpp -std=c++11
```

MPI — Message Passing Interface

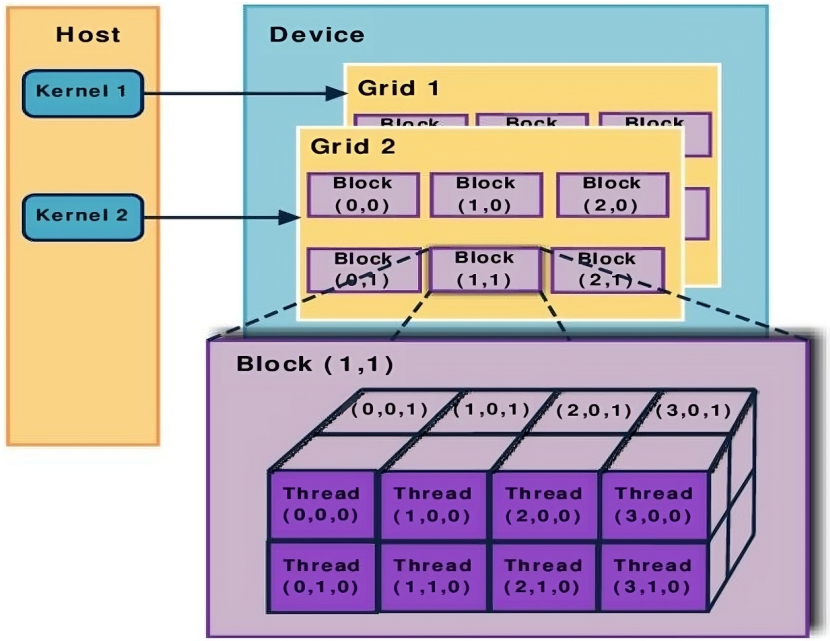
- multiple separated instances of program
- communication via messages
- memory of individual programs is separated (distributed) and data needs to be passed using messages
- distributed as separate library (MPICH, OpenMPI, IntelMPI)
- each process has to go until the end (finalize)
- works with C, C++, FORTRAN

MPI Program

```
#include <mpi.h>

MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
MPI_Comm_rank(MPI_COMM_WORLD, &mpi_rank);

... some code ...

MPI_Gather(&localPi, 1, MPI_DOUBLE, localPiArray, 1,
    MPI_DOUBLE, 0, MPI_COMM_WORLD);
if (mpi_rank==0){ ... }
MPI_Finalize();
```

Compute Unified Device Architecture (CUDA)

- proprietary and closed source parallel computing platform
- similar to C programming
- memory is separated from CPU
- tasks run on GPU are called kernels
- many low power cores



CUDA program

```
#include <cuda_runtime_api.h>

__global__ void pi_cuda(const double d, const int sizeN,
    char *res) { ... }

int main(){
...
cudaMalloc((void**)&resDev, sizeCube * sizeof(char));
grid.x = grid.y = grid.z = (sizeN / 8);
block.x = block.y = block.z = 8;
...
pi_cuda<<<grid,block>>>(delta,sizeN,resDev);
cudaMemcpy(res, resDev, sizeCube * sizeof(char),
    cudaMemcpyDeviceToHost);
...
```

Parallel Programming Competition

- One person can participate either individually or as a member of a team (not both), not more than 3 persons per team.
- To participate, one has to send e-mail to the address `hybrilit@jinr.ru` containing in attachment source code of proposed solution and inside the text body following information:
  - Name of the team + member(s) name
  - How to compile the proposed solution (Makefile)
  - How to run the proposed solution (run.sh)
  - All modules used have to be listed as well
- Competition deadline is today, 18:00

Rules

- Propose and implement the fastest method to calculate an approximate value of π via volume of 3D sphere obtained as a sum of elementary volumes. You may not use predefined value of π in any way except for estimation of error.
- Difference between approximated value and constant M_PI ("exact" value of π) must be less than 10^{-6}.
- Send program to the address `hybrilit@jinr.ru`
- The programs will be evaluated in the following way:
  - Program will be compiled according the rules participant sent in the e-mail
  - Program will be run 5 times, best and worst times will be removed and middle 3 runs will be averaged
  - The result will be checked on precision
- Extra task: Try to determine the size of grid (number of points), for which the approximation of π is most precise for given time limit.