Checkpoints: what, why and how

Code: https://staff.sharcnet.ca/guanw/2025/coco.tar.gz

Weiguang Guan, guanw@sharcnet.ca
SHARCNet/Digital Research Alliance of Canada

What and why

- What is checkpointing?
- Why do we need it?

A long job may fail in the course of execution.

- Previous valuable computation would be lost without checkpointing
- A Checkpoint allows the job to resume from an execution point where the checkpoint was saved.

Several causes of termination of a job

- When there is a system glitch
- When we didn’t allocate sufficient time for a job
- When there is a bug (e.g., running out of memory at a particular iteration)

Goal of the lecture

- General guideline in Wiki https://docs.alliancecan.ca/wiki/Points_de_contr%C3%B4le/en
- Concrete examples showing how checkpointing is implemented
    - Templates
    - Adapted to be used in a particular computation job

Checkpointing in different settings

Job types:

- Serial
- OpenMP
- MPI

Programming languages:

- C
- Python

Checkpoint file types:

- Plain text file
- JSon file
- Binary (even compressed) file (such as pickle)

Checkpointing not discussed in this talk

- Non-iterative computation jobs
- Some software packages have their own checkpointing mechanism (like Tensorflow)

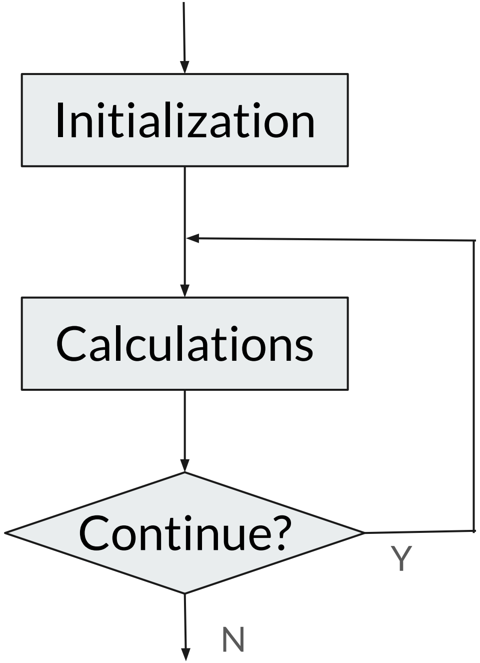
Course plan

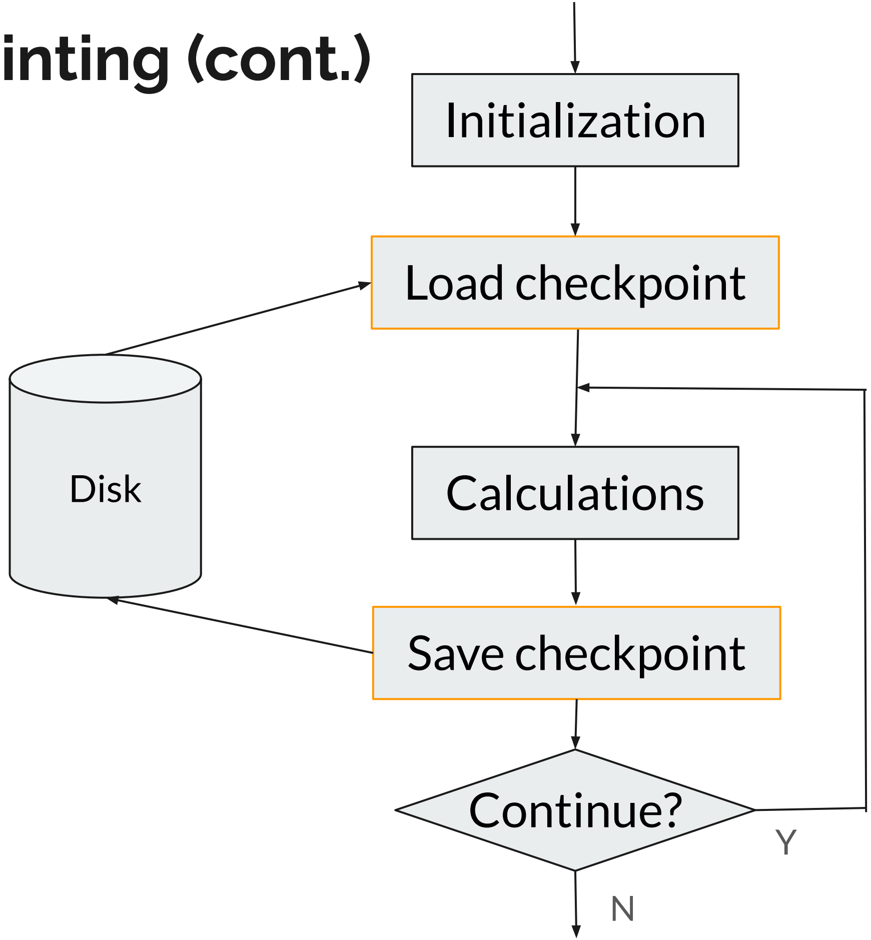
- State of an iterative process
- Save/restore a snapshot of state
- Frequency of saving checkpoints
  - Trade-off between overhead in saving checkpoints and the amount of computation that is about to lose
- Checkpoint files
  - Ascii file vs binary file
  - Single checkpoint (overwritten) vs multiple ones

How to implement checkpointing

- Explicit iterative process (for loop, while loop, etc)
- Implicit iterative process (one-line function call that encapsulates all the iterations, e.g., tf.keras.Model.fit(...))
  - Callbacks

How to implement checkpointing (cont.)

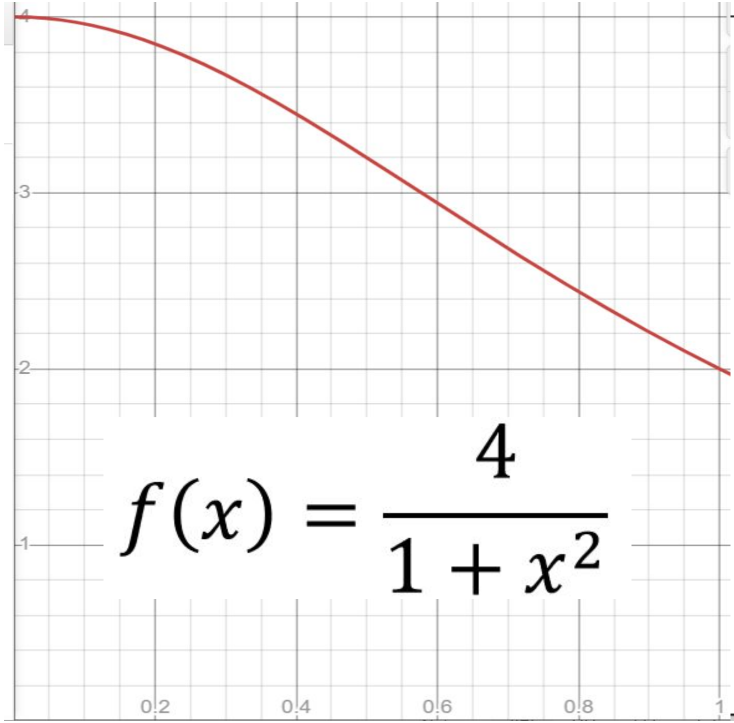




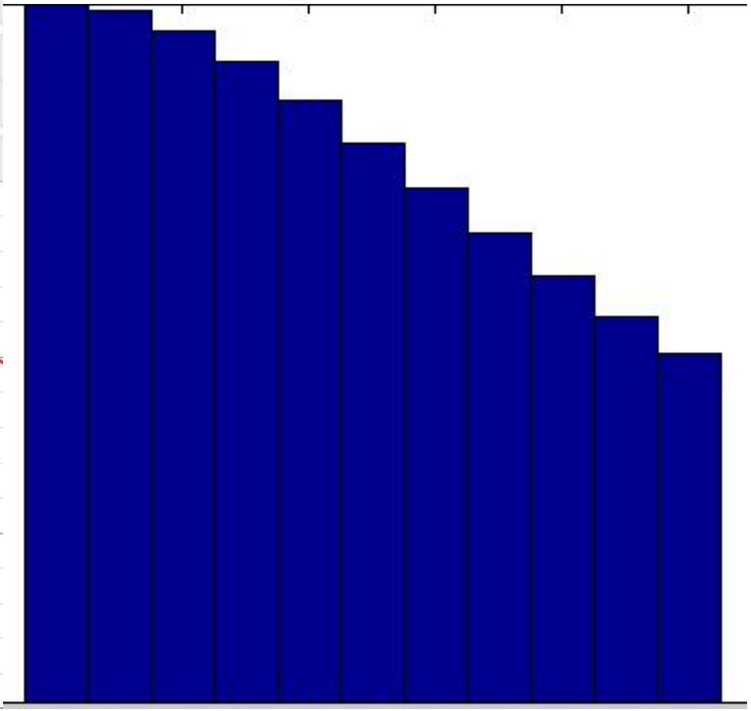
First example: Computing π value

$$\pi = \int_0^1 f(x)dx = \int_0^1 \frac{4}{1+x^2}dx$$

N = 10 bins
step = 1/10 = 0.1
area = f(0)*step + f(1)*step + ... + f(9)*step
= (f(0) + f(1) + ... + f(9)) * step



$$f(x) = \frac{4}{1+x^2}$$



First example: Computing π value (cont.)

State of iterative computation:

- Current iteration number
- Current summation of bin heights

First example: Computing π value (cont.)

Implementing checkpoints in

- Serial job
- OpenMP job
- MPI job

OpenMP implementation

Adding OpenMP directives

```
#pragma omp parallel
{
    #pragma omp for private(x) reduction(+:sum) schedule(runtime)
    for (int i=0; i < NUM_STEPS; ++i) {
        x = (i+0.5)*step;
        sum += 4.0/(1.0+x*x);
    }

    #pragma omp master
    {
        pi = step*sum;
    }
}
```

Covert a single loop to a nested loop

```
for (int i=0; i < 50000; ++i) {
    ......
}
```

```
for (int j=0; j < 50; ++j) {
    for (int i=j*1000; i < (j+1)*1000; ++i) {
        ......
    }
}
```

MPI implementation

```
for (int i=myid; i < NUM_STEPS; i += nprocs) {
    x = (i+0.5)*step;
    sum = sum + 4.0/(1.0+x*x);
}
```

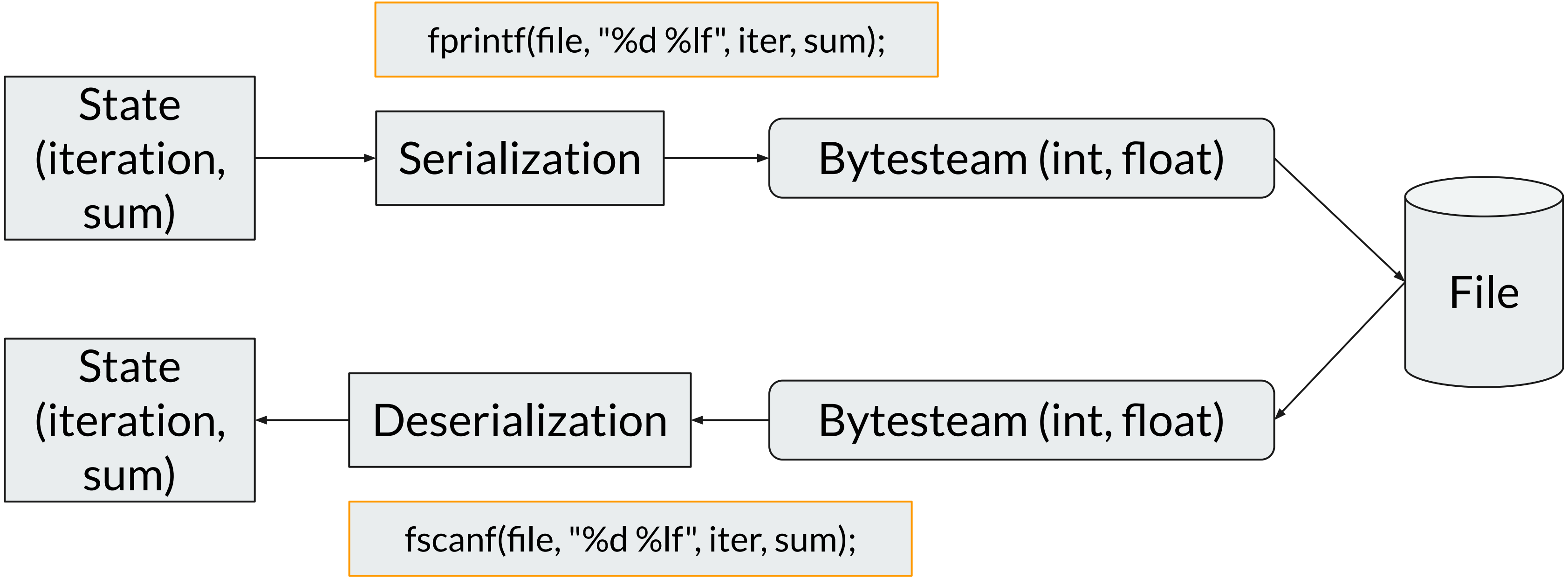
`nprocs`: number of processes

`myid`: id of current process

0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	...

nprocs=3, myid = 0, 1, 2

Summary of the first example



fprintf(file, "%d %lf", iter, sum);

State (iteration, sum)

Serialization

Bytesteam (int, float)

File

State (iteration, sum)

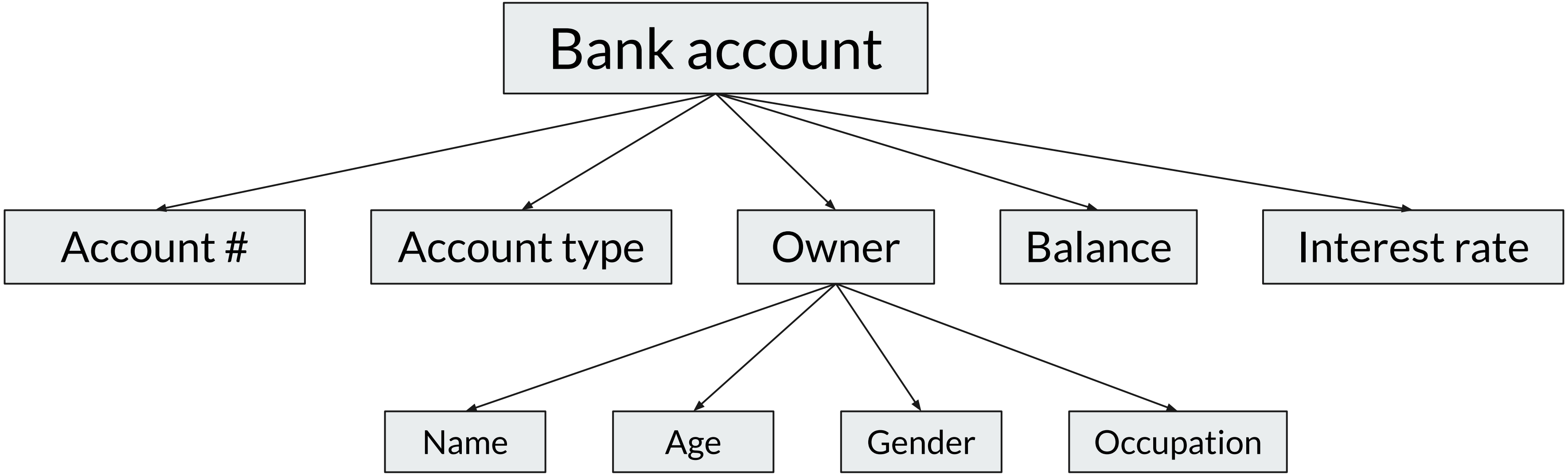
Deserialization

Bytesteam (int, float)

fscanf(file, "%d %lf", iter, sum);

Data serialization/deserialization

What if the data structure is too complex to decomposed into basic data types?



Second example: Generating Fibonacci sequence

Fibonacci sequence:

0	1	1	2	3	5	8	13	21	34	55	89	…
0	1	2	3	4	5	6	7	8	9	10	11	…

State: start position, last two numbers of the sequence

Initially, start = 2, last_two = [0, 1]

Second example: Generating Fibonacci sequence

Save/load checkpoints in following file format

- Plain text
- Json
- pickle

Some discussions about checkpoints

- Text vs binary vs compressed format
- Single vs multiple checkpoints
- Checkpointing frequency
- What if a checkpoint is corrupted

Frequency of saving checkpoints

- Saving checkpoints takes time, which depends on the size of checkpoint
- Choose an appropriate saving frequency
  - Higher saving frequency (or shorter interval between consecutive checkpoints) means more overhead
  - Lower saving frequency (or longer interval between consecutive checkpoints) means more loss if need to restore from a checkpoint.
- Keep only the latest checkpoint or a series of checkpoints

What if checkpoints are corrupted

- What can cause checkpoints corrupted
  - System issues
  - Job timeout
- How to detect corrupted checkpoints
  - Manual inspection if checkpoints are text files
  - Check sanity of checkpoints in your code
- What can we do if a checkpoint is corrupted