DM of APP

Nicolas Bouton

April 2021

Contents

NOTE: There are certainly many mistake.

1 MPI

1.1 Explaination

I commented a lot on my code. But I let explain my reasoning.
I decompose the array in each MPI process where each process have at least **SIZE_OF_ARRAY / NB_OF_MPI_PROCESS** and the

remaining is added to the **REMAIN_VALUE** first processes. It is the optimal way to decompose an array.
Each process will do its **local reduce** and then call **MPI_Reduce** to reduce all **local reduce** of MPI processes.
Therefore, the main process (Process 0) will have the result. I don't do an Allreduce because we don't need this. We just need one process to print the result. So I decide to use **MPI_Reduce** instead of **MPI_Allreduce**.

2 pthread

2.1 Explaination

I think my comment are enough but I will explain my reasonning.
I decompose the array to reduce in **NB_THREADS** part and for the remain I code the optimal way. I added one element for the first **value of remain** theads.
Then all thread do the reduce of the share array but with their part and after they will enter in the mutex, one at a time, for update the share reduce varaible with their local recude variable. (addition)
Then the main have the result because is it that create the shared reduce variable.

2.2 Value

2.2.1 Array

Reach	Attribute	Variable	Thread 0	Thread 1	Thread
-		-	0x7f4cb0ae1640	0x7f4cafadf640	0x7f4cb02e06
global	-	a	0x55c21e39108c	0x55c21e39108c	0x55c21e3910
global	__thread	b	0x7f4cb0ae1638	0x7f4cafadf638	0x7f4cb02e06
local of thread	static	c	0x55c21e391090	0x55c21e391090	0x55c21e3910
local of thread	-	d	0x7f4cb0ae0e44	0x7f4cafadee44	0x7f4cb02dfe
local of thread	static __thread	e	0x7f4cb0ae163c	0x7f4cafadf63c	0x7f4cb02e06

2.2.2 Explaination

1. Variable a

For the variable **a**, it is declare **global** without attribute, then all thread which are created will use the same address of the variable (they will don't create another). Typically, the main thread will create the variable and each thread created by the main thread will be used this variable.

Thus we have for the variable **a** the same value for all thread, also for the main thread (the processus).

2. Variable b

   For the variable **b**, it is declare **global** but with the attribute **__thread** which indicate to create another variable for each thread.

   Therefore, we have a different address for each thread, also for the main thread (the processus) because it is a processus and not a thread create by **pthread**.

3. Variable c

   For the variable **c**, it is declare **locally in the thread function** with the attribute **static** which indicate that variable is initialized only once by the program.

   Therefore, all thread use the same variable and we have the same address for each thread.

4. Variable d

   For the variable **d**, it is declare **locally** in the thread function withou attribue.

   Therefore it is local and all thread have different vairable. So all address differ.

5. Variable e

   For the variable **e**, it is declare **locally in the thread function** but with **two attribute**, **static** and **__thread**. The first indicate that the variable is initialized only once and the second indicate that all thread have a different variable. So each thread will initialazed once our variable.

   Therefore, as each thread have their own variable, all address differ.

3 OpenMP

3.1 Explanation

For openmp, David Dureau said us that is a good way to put by default all variable private with `default(none)`.
On the code we just need to have the **for** in omp parallel, so I put all **omp** attribute in the same line.
To decompose the array we need to have a private index for all thread, so **i** is private. And we want to make a reduction, so we make a reduction on variable **sum**.

3.2 Value

In **omp parallel** clause, we don't specify the default "vision", therefore all varaible is shared by default and I suppose that information is knowed for the following explanation.

3.2.1 Array

Reach	Attribute	Variable	Thread 0	Thread 1	Ma
global	-	a	0x5577fa58906c	0x5577fa58906c	0x5577fa5890
global	#pragma threadprivate	b	0x7f9225b78d7c	0x7f9225b7763c	0x7f9225b78d
local	static	c	0x5577fa589070	0x5577fa589070	0x5577fa5890
local	-	d	0x7ffca463cb28	0x7ffca463cb28	0x7ffca463cb
local	#pragma omp private	e	0x7ffca463cad4	0x7f9225b76d84	0x7ffca463cb
local	#pragma omp shared	f	0x7ffca463cb2c	0x7ffca463cb2c	0x7ffca463cb

3.2.2 Explanation

1. Variable a

   The variable b is declare **global** in the code and have not attribute.

   Therefore, all thread will have the same variable and the same address.

   And the main have the same address of **thread 0** because **main** is the **thread 0**.

2. Variable b

   For the variable b, the line was not correct because **threadprivate** is a directive of omp and you don't specify omp after the pragma. So I added it.

   The attibute **threadprivate** specify that all thread will create their own variable.

   Therefore, all thread will have the same variable and the same address.

   And the main have the same address of **thread 0** because **main** is the **thread 0**.

3. Variable c

   For the variable c, it is declare locally in the function and with attribute **static**, so tha variable will be initialazed once in the program and each thread will have the same value because on **omp parallel** clause we don't change his attribute.

   Therefore, all thread will have the same variable and the same address.

   And the main have the same address of **thread 0** because **main** is the **thread 0**.

4. Variable d

   For the variable d, it is declare locally in the function, without omp attribute in **omp parallel**.

   Therefore the variable is shared and all **thread** will have the same variable. (the same address)

   And the main have the same address of **thread 0** because **main** is the **thread 0**.

5. Variable e

   For the variable e, it is declare locally in the function but on **omp parallel** clause we specify that is **private** and each thread will create their own variable. Even the **thread 0** will create another variable. It is for thsi that the address differ between **main** and **thread 0**.

6. Variable f

   For the variable f, it is declare locally in the function but on **omp parallel** clause we specify that is **shared**. Therefore all thread will shared the same variable created by the main.

   Therefore we have the same adress for all thread and main.

4 Reference

Github repository: `https://github.com/Sholde/dmapp`
If you are intersted, I developped a tool for another module, named TOP, where we had to debug a MPI application and optimize it. So I develop a **MPI Profiler**:
Link: `https://github.com/Sholde/mprof`