Outlook

# Re: Split Inference on Networked Microcontrollers

**From** Junyu Lu <thisislujy@163.com>
**Date** Thu 19/09/2024 08:55
**To** Shashwath Suresh <S.Suresh-3@student.tudelft.nl>

Dear Shashwath,

I forgot to mention, between downloading worker data and coordinator data, you need to reset the MCU (power off and on, or press the reset button) to reset the variables. So the correct steps should be

1. Put the MCU into download mode
2. Write 'c' through serial, set the "file" ('write_into_mcus.py', line 30) to Coordinator (and update 'd_1' based on which MCU I am setting up)
3. Run write_into_mcus.py
4.Reset the MCU
5....
6....
7....
8....

Best,
Junyu

---- Replied Message ----

| | |
|---|---|
| From | Shashwath Suresh<s.suresh-3@student.tudelft.nl> |
| Date | 09/19/2024 01:47 |
| To | Junyu Lu<thisislujy@163.com> |
| Subject | Re: Split Inference on Networked Microcontrollers |

Dear Junyu,

Thanks a lot for the detailed explanation! Now that you mentioned I misunderstood the role of the coordinator, I want to be 100% sure that I am loading the weights and mapping information correctly.

So, the file to write (write_into_mcus.py, line 30) is set to always be Coordinator.json. Which means I will have to change this once I load the coordinator information, so that the weights are also loaded. Furthermore, the storing of weights only happens when I enable logging through serial, so I need to manually select coordinator and datalog to ensure all the information is loaded onto the MCU.

Based on the above, I think the sequence should be like this:

1) Put the MCU into download mode

2) Write 'c' through serial, set the "file" ('write_into_mcus.py', line 30) to Coordinator (and update 'd_1' based on which MCU I am setting up)
3) Run write_into_mcus.py
4) Write 's' through serial, set the "file"('write_into_mcus.py', line 30) to "worker_(num)"
5) Run write_into_mcus.py
6) Flash worker code with the correct IP, MAC and MCU ID
7) Run tcp.py

Can you confirm if this sequence of events is correct?

Once again, thanks a lot for helping out!

Best Regards,

Shashwath

On 18 Sep 2024, at 19:04, Junyu Lu <thisislujy@163.com> wrote:

Hi Shashwath,

So, basically, the 4 arrays are there to identify the number of inputs that each MCU should handle and the split points of the .bin files for every layer, with these arrays, each MCU knows when it has received all of its inputs, produced all of its results and how to decode from flash memory for each layer. As I mentioned in my readme file, these can be obtained while performing simulation, you can simply print the length of the inputs and the results during the inference of each layer.

For me, I obtained the length of the result by printing them during testing in "pc_code/Algorithms/src/main" line 608-610, the intput length is obtained by line 613 (which is commented out). I think you can find a better way to do this through simulation, if you wish to do what I did, you can simply rewrite the locations for your files, use dummy reference, and comment out the comparaing phase. Note that each MCU have different lengths, so you need to print them out seperately. For coordinator lines and worker lines, you should get them through the serial monitor after the download is complete, you can refer to filesys.h lines 217-224 and 438-445 for more information.

As for your questions of the procedure, I think you misunderstood the role of the coordinator when deployed on real hardware. For real implementation on hardware, because I only have 3 MCUs during that period, I partially followed the procedure described in figure 7.5 of my thesis where the role of the coordinator is merged into each worker MCU, so, MCU1 should have its worker file as well as the first portion of the coordinator file, you can refer to line 102 of the "write_into_mcus.py" file, here I wrote "data = d_1[0]" which means I am writing the first portion of the coordinator file to the first coordinator, so perhaps you need to change that in such a way "data = d_1[0] for the first worker ,data = d_1[1] for the second worker and so on". To explain further, because each MCU can have its own part of the coordinator file, which stored the information of which MCUs to send to after its calculation, it can simply send this data to the PC, and PC is only responsible for the synchronization of the process, so the 2 tasks of the coordinator is decomposed, with each MCU responsible for storing the knowledge and the PC is responsible for the synchronization, if you wish to have one MCU specifically for coordinator, you may need to redesign the code for downloading and the "tcp.py" file as well as the worker code.

Let me know if this email solve your problem, if you have any more questions, please let me know.

Best regards,

Junyu

At 2024-09-18 19:59:47, "Shashwath Suresh" <S.Suresh-3@student.tudelft.nl> wrote:

> Dear Junyu,
>
> Thank you for sharing your email with Hao.
>
> I am currently working on expanding your thesis work so that we can run inference on a bigger testbed. I have tried to recreate the experiment you have documented in your thesis with 3 MCUs, but I am unable to go beyond the first layer. I believe this has to do with the configuration of the setup, so I wanted to make sure that I have understood the configuration procedure correctly.
>
> Based on the provided readme and the thesis, here is an overview of what I have done to configure the setup.
>
> 1) Run simulation to obtain the weight distribution json files
> 2) Flash each MCU with download.ino
> 3) Based on whether the MCU is a worker/coordinator, download the correct .json file (generated by simulation) by logging the data into into a .bin file
> 4) Flash worker_code.ino into each microcontroller (after giving the correct IP, MCU ID and MAC addresses each time)
> 5) Run tcp.py to perform inference
>
> Is this set-up procedure correct or am I missing something?

On closely looking at the source code, I found out that there are 4 arrays that define the lengths of inputs, results, coordinator lines and worker lines. These are from the .txt files in the Lines folder, but I am unable to find how these have been generated. Could you let me know how I can obtain these values? I think if my setup procedure is correct, then they could explain why I am unable to perform inference.

Thanks in advance for your help.

Best Regards,

Shashwath Suresh