MPI Execution for Educational Support

Instructions

This is an MPI code execution tool with an educational purpose. The provision of computational resources is a challenge that can affect the understanding of parallel programming concepts. With this in mind, this application aims to address this problem by providing computer science students with virtual machines that parallelize the execution of MPI programs.
To ensure the correct execution of programs, some precautions are necessary when uploading the code. Below is a functional example for your first test of the application: copy and paste the following code into files named 'Makefile' and 'mpi_hello.c', upload these files, and evaluate the result.

-> Information regarding the files:
Be careful when constructing the Makefile. It needs to be correctly formatted to function properly. All compilation and execution commands must be in the 'run' rule. Additionally, do not change the name of the file for writing results; it must be called 'result.txt'.

In the 'run' rule, you can choose the number of processes to be executed and which hosts will run these processes. The host options are worker1, worker2, and worker3. Each host is a virtual machine at your disposal.
Examples of choices:
'mpirun -np 4 -hosts worker1 ./$(PROGRAM) >> $(RESULT)'
'mpirun -np 3 -hosts worker1, worker2, worker3 ./$(PROGRAM) >> $(RESULT)'
'mpirun -np 5 -hosts worker1, worker2 ./$(PROGRAM) >> $(RESULT)'

Remember not to change the name of the result.txt file.

Note: To avoid a very high execution load, it is recommended that you test your code before submitting it to the application.

Makefile

```
# Name of your MPI program
PROGRAM = mpi_hello

# Your MPI C compiler
MPICC = mpicc

#DO NOT CHANGE
RESULT = result.txt

# The program file. Note: If you have multiple source files, add them here
SRCS = mpi_hello.c

# Rules
all: $(PROGRAM)

$(PROGRAM): $(SRCS)
	$(MPICC) -o $(PROGRAM) $(SRCS)

run: $(PROGRAM)
	mpirun -np 4 -hosts worker1,worker2 ./$(PROGRAM) >> $(RESULT)
clean:
	rm -f $(PROGRAM)
```

mpi_hello.c

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
    int rank, size, namelen;
    char hostname[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Get_processor_name(hostname, &namelen);

    printf("Hello, world! I am process %d of %d running on %s\n", rank, size,
hostname);

    MPI_Finalize();
    return 0;
}
```

Upload Files... Drop files here

Send Files Download Result