MCCK User Manual

Version 1.0

Mathematics and Computer Science Division
Argonne National Laboratory

Kyle Gerard Felker

Andrew Robert Siegel

April 25, 2013

Contents

1 Introduction

MCCK, or Monte Carlo Communication Kernel, is a highly-scalable, portable C based library for simulating Monte Carlo neutron transport environments in nuclear reactor analysis. MCCK employs domain decomposition and the Message Passing Interface (MPI) to simulate the computation of fission cycles of massive numbers of neutral particles in parallel. The active development of MCCK occurs on the Subversion repository, located at

`https://svn.mcs.anl.gov/repos/cesar-codes/mini_apps/mcck`.

Access is limited to approved users with MCS domain accounts.

2 Background

Computational scientists have long used PDE-based deterministic methods for nuclear reactor analysis. The stochastic Monte Carlo (MC) approach offers several potential advantages compared to the deterministic formulations, and arguably extrapolates more readily to the modern hardware trends that lie on the path to exascale computing. MC simulations, though, are notoriously expensive computationally and require a tremendous amount of computing to obtain converged statistics. A number of algorithmic and implementation challenges thus remain before they can be robustly applied to practical reactor analysis.

One of these challenges involves developing memory decomposition strategies to enable robust Light Water Reactor (LWR) simulations. The key issue is that for traditional serial or naturally parallel MC implementationsi.e., those based on replication of the global domain on each node neutrons can independently be created and tracked one by one from creation to absorption. This approach, however, severely limits available global memory to the maximum available on an individual node. A significant amount of the memory per node is needed to tally neutron interactions for the pre-defined geometric regions, or tally regions. For many classes of calculations such an approach is adequate; geometric descriptions are simple and the number of tally regions is modest. However, for real-world reactor core simulations, the need to tally localized quantities for hundreds of isotopes across tens of thousands of bins results in a memory requirement for tallies alone of approximately 10^{12} bytes of memory.

One approach to increasing available memory is to implement spatial domain decomposition for the tallies. When carrying out physical space domain decomposition, though, tracking particles one by one from birth to absorption is no longer feasible performance-wise. Each process owns a subset of the physical space domain and only tracks the particles that pass within its boundaries. Good performance then requires tracking the particles in stages that is, moving all particles locally on a partition until they are absorbed or reach the partition boundary. At the end of each stage large blocks of particles need to be moved to adjacent partitions [1].

The goal of the mini-application is to gauge the relevant performance regimes of the spatial domain decomposition technique and evaluate the benefits of various memory redistribution techniques.

3 Getting Started

3.1 Installation

1. Download the latest stable version of MCCK can be downloaded

   `https://cesar.mcs.anl.gov/content/software/neutronics`

   Once downloaded, you can decompress MCCK using the following command on a linux or Mac OSX system: `tar -zxvf mcck-1.0.tgz` This will create the `mcck-1.0/` directory. The source code is found in `mcck-1.0/src/`

2. (Optional for MADRE or FPMPI logging) MCCK can automatically link the MADRE and FPMPI libraries, but it presently lacks a configuration script. Therefore, if these features are desired, you must manually enter the locations of these libraries. Change into the `trunk` directory. Edit the `Makefile.h` file to accurately reflect the locations of your MADRE and FPMPI source directories.

3. After ensuring that your desired C/MPI compiler is correctly defined in `Makefile.h` "MPICC" variable in the top level directory, type `make`. You have a few options when completing this step, all accessed by defining the corresponding macro as anything on the command line. For example, "`make INCLUDE_MADRE=1`"

IMPORTANT NOTE: You must `make clean` when rebuilding MCCK with a new option, as the GNU Make utility won't automatically recognize changes in the macro definitions

- INCLUDE_MADRE=1
  Defines the INCLUDE_MADRE macro in the `MC.h` header file which provides the `MC_MADRE` option for communication. Links the MADRE library.
- LOG_FPMPI=1
  Links the FPMPI library with MCCK to produce a text file containing timings of communication after runtime.
- LOG_MPE=1
  Links the MPE library with MCCK to produce a Jumpshot log file for visual logging. Cannot use this flag simultaneously with LOG_FPMPI.

4. Change into the `test` directory.

If compilation completes successfully, the executable "Main" will be placed in the `main/` subdirectory.

To create your own MCCK Monte Carlo simulator, include "MC.h" in your source code file and link the library.

3.2 Running Jobs

To run MCCK with default settings, use the following command:

```
mpiexec [-np NUM PROC] ./Main NPARTICLES GLOBAL_LEAKAGE
```

This command launches an initial batch of `NPARTICLES` on each of the `NUM PROC`. The two arguments are required for all simulations. There is no restriction on the number of processors for the simulation (tests have been executed with over 16,000 processors). The leakage rate, λ, is a lumped parameter which represents the neutron absorptive processes in an actual physics code.

$(1-\lambda_i)$ is the average probability that a particle is absorbed on processor i's physical domain, and λ_i is the average probability that a particle continues to leave that processor's domain. Thus, at each stage, the non-absorbed particles on each process are buffered in preparation for movement to the new processor's domain.

At each stage, the code will print to STDOUT a line similar to:

stage 1, 160000 total np ([min: 40000 max: 40000 mean:40000.00 delta:0.00])

⋮

stage 77, 52 total np ([min: 4 max: 19 mean: 13.00 delta:6.00])

until all of the particles have been absorbed and the simulation ends. The statistics detail the communication stage number, the total number of live particles, the smallest number of live particles on a processor, the largest number of live particles on a processor, the mean, and "load imbalance" factor (i.e. difference between the mean and the max).

3.3 Runtime options

In addition to the standard runtime arguments, the user has several optional arguments at his or her disposal.

- Strict load balancing (-m strict/nostrict)

  Strict load balancing ensures that every processor has an identical number of live particles at each stage. `nostrict` is the default.

- Boundary conditions (-b BNDRY_REFLECT/BNDRY_LEAK/BNDRY_PERIODIC)

  For the cubic domain, when neutrons travel to the boundary, they can either bounce back, leave the simulation, or wrap around to the opposite side. These are the respective options. `BNDRY_REFLECT` is the default.

- Constant random number seed (-r)

  By default, the `srand()` call in the code is seeded by the system time, thus creating a new pseudorandnom sequence on each execution. If you are trying to reproduce results, use this option to seed the generator with the constant 3333.3.

- Variable leakage file (-f FILE.txt)

By default, the leakage rate λ_i is constant across processors. However, one can create an input file formatted as:

```
6
10 .5
100 .9
10 .5
10 .5
10 .5
10 .75
```

where the first line is equal to NUM PROCS and each of the subsequent NUM PROCS lines has the NUM PARTICLES and LEAKAGE local to that processor.

References

[1] K. G. Felker, A. R. Siegel, and S. F. Siegel. Optimizing memory constrained environments in Monte Carlo nuclear reactor simulations. *International Journal of High Performance Computing Applications*, 2012.