Automated Parallel Execution of SCALE/MAVRIC Replica Simulations

Garrett Youngblood, Bojan Petrovic*
Georgia Institute of Technology
Nuclear and Radiological Engineering Program
Atlanta, GA 30332, USA
gyoungblood8@gatech.edu, bojan.petrovic@gatech.edu

ABSTRACT

The MAVRIC sequence of the SCALE code package for shielding analyses implements the CADIS/FW-CADIS methodology for automated variance reduction (VR) using a hybrid approach. Deterministic calculations generate VR parameters that are subsequently passed to the Monte Carlo code Monaco. While Monaco with VR typically achieves significant speedups, it is a serial code, i.e., it does not have the capability for parallel simulation, which leads to long wall-clock times. This could be addressed by manually submitting simultaneous multiple replica runs with different random number generator seeds. SCALE offers a utility for combining results of such simulations. This approach was historically called Poor Man's Parallel (PMP). The overhead (engineering time) needed to manually run parallel cases and then combine results is small to moderate, in particular for long simulations. However, it is error prone. Moreover, for shorter preliminary evaluations, or for testing optimum simulation strategies, it may become an impediment. To address this, a script called Poor Person's Parallel ("PPP", or "P-cubed") was developed. The script automates the necessary steps including: generating multiple copies of input; modifying the random number seed of each replica run; spawning parallel runs; and, collecting/combining results. This enables quick and error free submission of many test/scoping simulations in early phases of a project, or, easily submitting many parallel reference runs for final long simulations. Two versions of the script were developed and tested: one for Windows and one for Linux. The Windows version is aimed at Senior Design class students, that frequently run SCALE simulations on their laptops. While typical laptops have no more than 8 cores, and the optimum number of parallel runs is almost always less than the number of cores, P-cubed can still provide a non-trivial speedup by a factor close to the number of cores. Also, since the runs are seamlessly submitted and run, it is easy to test and determine the optimum number of runs for each specific problem and computer configuration. Linux version provides similar functionality, but with a potential for higher speedup as Linux nodes/clusters have more cores. This paper describes the P-cubed script and presents results from its use on both Windows and Linux machines to perform multiple runs in parallel.

Keywords: *Shielding analysis, SCALE, MAVRIC, variance reduction, poor man's parallel*

1 INTRODUCTION

Historically, shielding calculations relied either on deterministic methods, which involved approximations and substantial memory usage due to phase-space discretization, or on Monte Carlo methods, which often result in extended simulation times [1], [2]. In recent years, hybrid approaches that integrate the benefits of both techniques have emerged as the most effective solution. While many hybrid techniques have concentrated on optimizing results for specific, localized areas, the FW-CADIS method, implemented within the MAVRIC sequence of the

SCALE6 package, allows for the concurrent variance reduction (VR) of Monte Carlo simulations across large regions or even the entire problem domain.

The SCALE (Standardized Computer Analyses for Licensing Evaluation) code system [3], developed and managed by Oak Ridge National Laboratory (ORNL), is a well-established suite for modelling and simulating nuclear engineering processes, widely used in both analysis and design. MAVRIC (Monaco with Automated Variance Reduction using Importance Calculations) is a sequence within the SCALE6 code system for performing shielding calculations [4]. As its name implies, it makes use of Monaco, a fixed-source Monte Carlo shielding code [5], and a hybrid deterministic-stochastic method to automatically generate VR parameters for Monte Carlo simulations. The details of this methodology are discussed in more detail elsewhere [7], [8], [9].

The integration of Monaco and VR within MAVRIC has the potential to significantly decrease the computational time required for Monte Carlo simulations, depending on the specific problem. However, since Monaco operates as a serial code, the wall-clock times can still be substantial-to-prohibitive for challenging shielding problems. The goal of this research is to explore the application of a scripting technique to automate the PMP method of pseudo-parallelization and examines its impact on Monaco's performance under both Windows and Linux operating systems.

2 METHODS

The effect of the P-cubed script on Monaco performance was evaluated on two distinct computing systems: a Dell 3600 Workstation equipped with 12th Gen Intel® Core™ i9-12900K (16 cores, 24 threads) and 64 GB of RAM, and an HP Envy x360 Laptop featuring 8 CPUs and 16 GB of RAM. These computers were selected based on their availability and their respective operating systems. It is important to note that the P-cubed script is compatible with a wide range of computing environments, including individual computers and large-scale computing clusters.

2.1 Test Problem

The primary model used in this study was the TN24-P spent fuel cask from the SCALE 6.3.1 User Manual [9]. This problem was selected because calculating the dose rate with reasonably reduced statistical uncertainty across the entire problem domain would be exceptionally challenging in the absence of MAVRIC. Therefore, it was deemed a representative case for testing the performance of MAVRIC. The problem geometry is shown in Figure 2-1. Note, the number of particles per generation was increased from the original 15,000 to 2.5 million on the Windows system and 5 million on the Linux system. This adjustment was made to extend the problem's runtime and minimize statistical variation in the measured performance of parallel Monaco simulations.

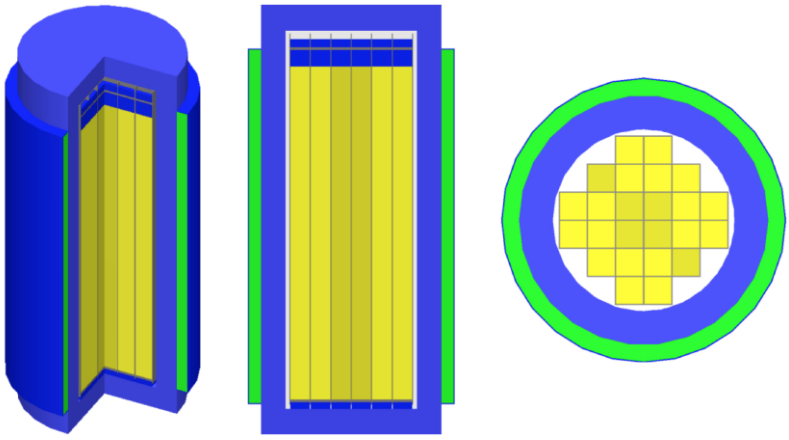

Figure 2-1: MAVRIC model of the TN24-P cask. [9]

2.2 P-cubed Script

Two distinct versions of the script were developed to automate the PMP approach for running MAVRIC simulations in parallel: one written in *batch* for Windows systems and the other in *bash* for Linux systems. Despite the difference in scripting languages, the algorithms in both scripts are functionally equivalent. The process begins with the user providing the following input: the name of the problem's input file and the number of parallel runs desired. Subsequently, copies of the problem's input file are created, one for each parallel run, with each file assigned a unique 16-digit hexadecimal random number seed. The script then initiates all simulations by repeatedly invoking *scalerte* for each input file, with a brief 3-second delay between each run to mitigate potential file access conflicts, particularly related to cross-section libraries.

After the final input file has finished running, a new input file is generated that invokes SCALE's mtAverager utility, which combines the outputs from the individual runs by averaging them, weighted by the number of histories in each run. Finally, the algorithm calls *scalerte* to execute this new input file, quickly producing an output containing the combined results from all parallel runs. A general overview of the script's algorithm is provided in Figure 2-2 for clarity.

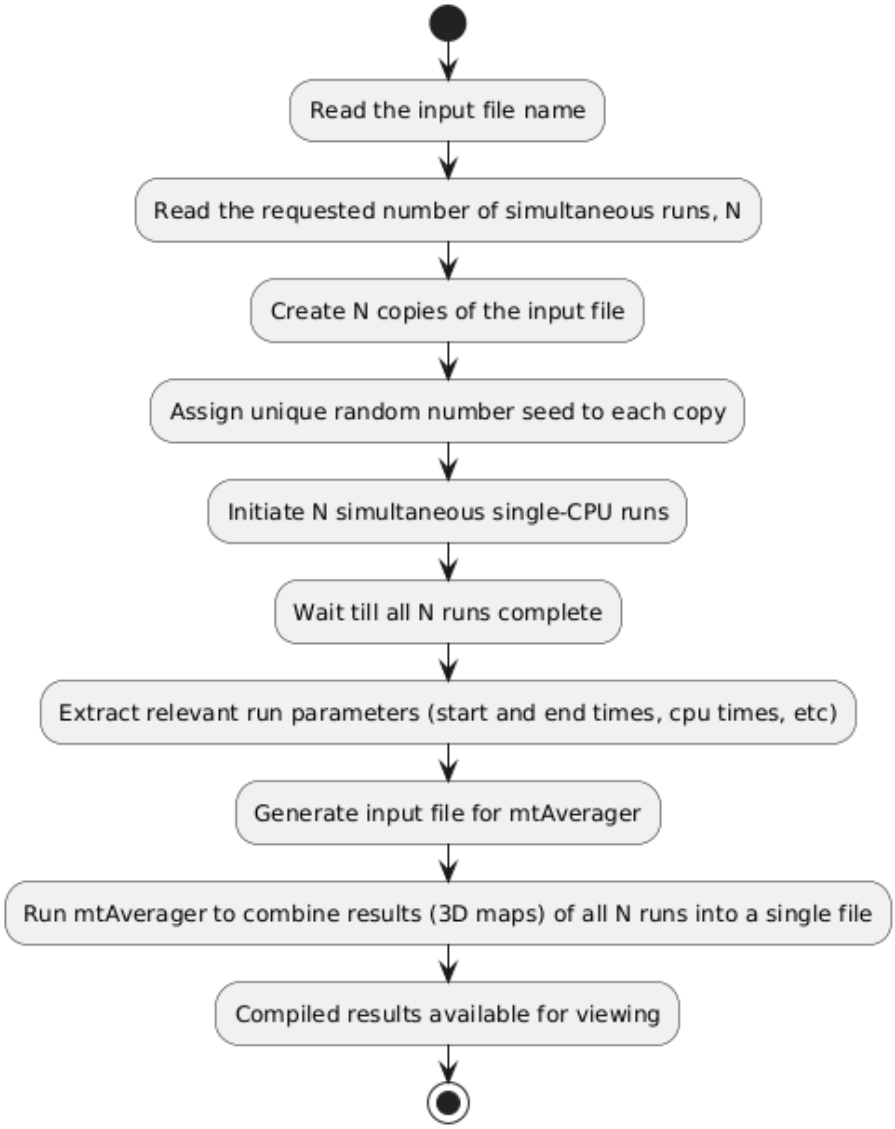

Figure 2-2: P-cubed script algorithm flowchart

3 RESULTS & ANALYSIS

3.1 Windows System

The P-cubed script was evaluated on the Windows system using the TN24-P spent fuel cask problem, with 1, 2, 4, 6 and 8 parallel runs. The resulting runtime for each parallel set of Monaco simulations indicates that, as anticipated, the runtime per run increased as the number of parallel runs increased. This can be attributed to the higher CPU utilization, which led to increased competition for other computational resources, most likely I/O, thus resulting in longer average computation times [9]. Notably, as illustrated in Figure 3-1, the difference between the fastest and slowest Monaco runtimes was relatively minor, with a maximum variation of approximately 50 seconds, i.e., ~2%. This suggests that the parallel runs did not experience significant load imbalance.

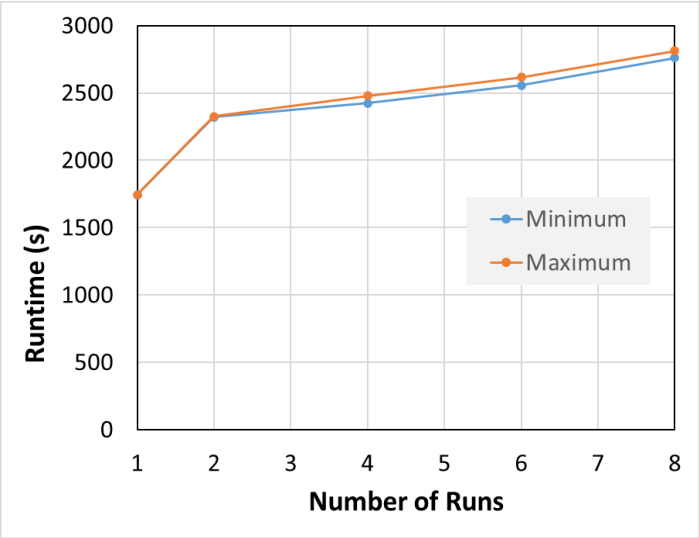

Figure 3-1: Minimum and maximum Monaco runtimes on Windows system.

From a practical standpoint, the application of the P-cubed script on the Windows system led to notable enhancements in Monaco performance. As shown in Figure 3-2, the speed-up factor exhibited an approximately linear improvement with the increase in the number of parallel runs, achieving a maximum speed-up of approximately 5 when all 8 CPUs were utilized.

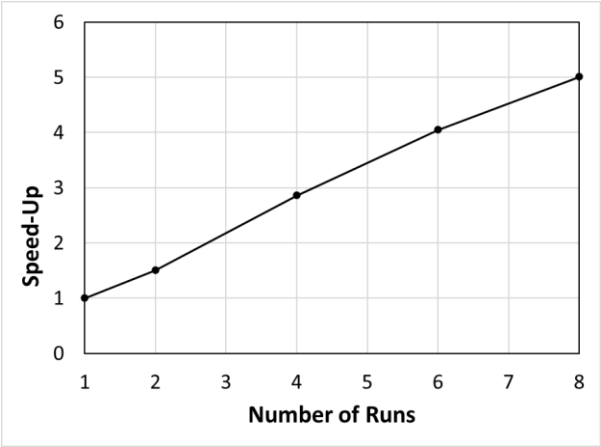

Figure 3-2: Monaco speed-up on Windows system.

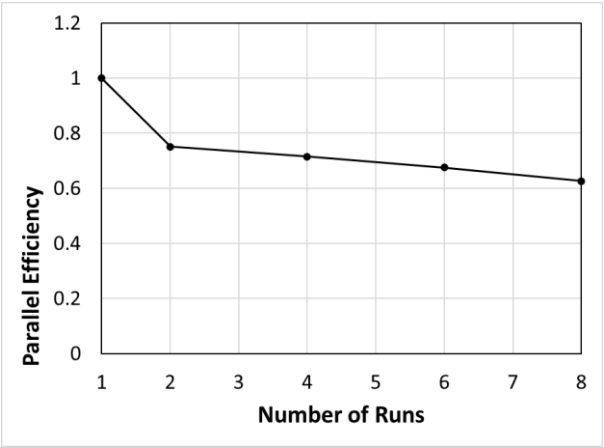

Figure 3-3: Monaco parallel efficiency on Windows system.

The parallel efficiency of simulations when using the P-cubed script was also considered as an indicator of the script's usefulness. On the Windows system, there was an initial decline in efficiency to approximately 75% as the number of CPUs increased from 1 to 2. However, as seen in Figure 3-3, beyond this point, the efficiency decreased slower, at a relatively constant rate, remaining a respectable about 63% when utilizing 8 CPUs.

The Windows version may be particularly beneficial for student projects, because students often don't have access to larger parallel clusters, but only to their own laptops. A speedup of five times could make a difference for demanding simulations in making them viable.

3.2 Linux System

The P-cubed script was also assessed on the Linux system using the same problem input file, with 1 to 32 parallel simulations. Similar to the Windows system, the Monaco runtime for each set of parallel simulations increased with the number of parallel runs; however, on the Linux system, the trend exhibited greater variability. Before aiming to interpret results, it is useful to recall that the Linux system had 16 cores with 24 threads, i.e., 8 single-thread cores and 8 double-thread cores. As shown in Figure 3-4, for up to 8 runs, the Monaco runtimes followed a trend comparable to that observed on the Windows system (Figure 3-1), increasing at an almost constant rate as the number of parallel runs grew, and with the minimum and maximum runtime being almost equal. However, at 10 CPUs the minimum and maximum runtime start to separate, grow more significantly, and the maximum runtime becomes significantly longer than the minimum runtime, up to 24 parallel simulations, when they again become similar and plateau. Beyond this point, both the minimum and maximum runtimes continued to rise sharply until plateauing at 24 runs. Between 24 and 32 runs there was little change in the average runtime. Explanation of this at first glance strange behaviour may be attempted by the specific architecture of the Dell workstation, with CPUs of two types (8 performance-cores allowing 16 threads and 8 Efficient-cores, allowing 8 threads, so called 8P+8E).

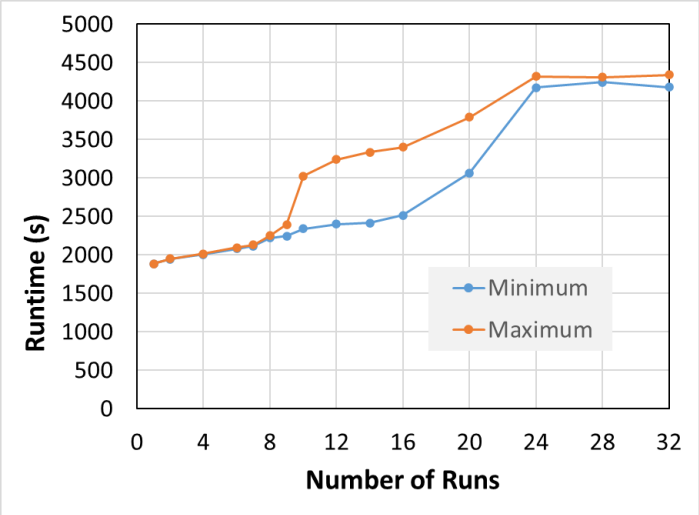

Figure 3-4: Minimum and maximum Monaco runtimes on Linux system.

We hypothesise that up to 8 parallel runs are executed one each on a separate performance core; there is no load imbalance, and the minimum and maximum runtimes are essentially identical. Beyond 8 threads, some are performed on performance-cores and some on efficient-cores, leading to disbalance and different execution speed. Beyond 16 runs, some runs are performed on a single-thread core, while some runs are performed in pairs as 2 threads on the same core. As a result, the maximum runtime almost doubles. As we reach 24 or more parallel runs, all threads are busy all the time, resulting in longer but even runtimes.

This is reflected in the speed-up and parallel performance depicted in **Error! Reference source not found.** and **Error! Reference source not found.**, respectively. Up to 8 runs the speed-up is almost linear, amounting to 6.8 for 8 runs. It slows down beyond 8 runs, but it still reaches 10.3 for 16 runs. Beyond that, more runs activate additional threads, but still use the same 16 cores, leading to a load imbalance and stagnation of speed-up up to 24 runs. Simultaneous execution of 24 or more runs leads to full utilization of the system all the time and the speed-up factor increases again, up to 14.1 for 32 runs. Due to memory limitation it was not possible to test beyond 32 runs, but it is expected that the speed-up would plateau again.

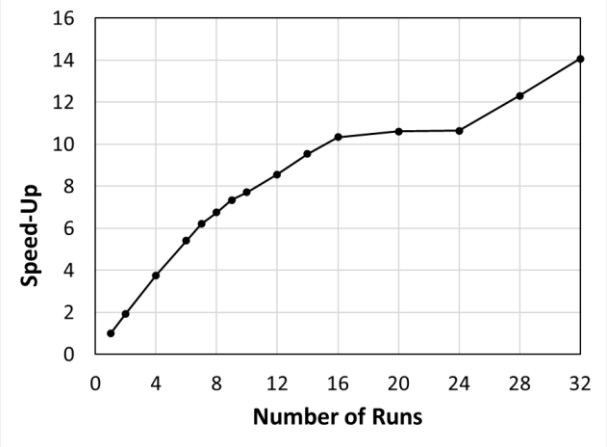

Figure 3-5: Monaco speed-up on Windows system.

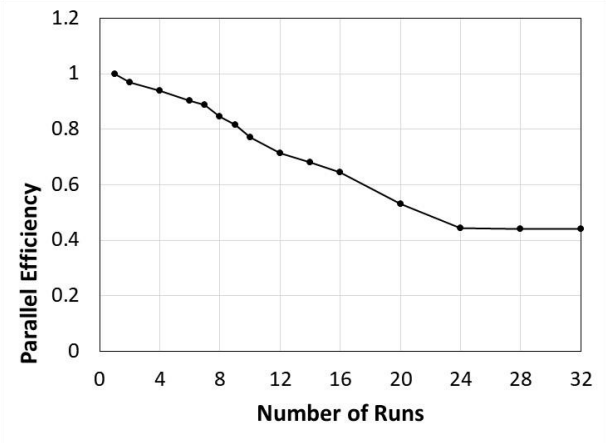

Figure 3-6: Monaco parallel efficiency on Windows system.

Despite the more complex runtime behaviour, the application of the P-cubed script on the Linux system resulted in a greater performance improvements compared to the Windows system, as expected due to more cores as well as more robust internal resources. At 8 CPUs, the Linux system achieved a speed-up of approximately 6.8, as compared to ~5 on Windows. The maximum speed-up on Linux of ~14 for 32 runs is in fact fairly good considering that the system has only 16 physical cores.

3.3 On-going and Future Work

Short of using system profiling tools, it would not be possible to directly confirm the hypothesis explaining the complex behaviour observed on the Linux workstation shown in Figure 3-4. Therefore, a similar set of parallel runs was performed for another, using a simpler problem also from the test set provided with SCALE. A very similar behaviour was observed, further supporting the explanation. However, to keep the paper compact, we are not presenting those results, since not much new information would be obtained.

Of more interest is testing the effectiveness of the P-squared script on larger computers, more representative of high performance computing centres. Initial tests have been performed on a larger Linux cluster with up to 48 cores per node. Preliminary results indicate good performance of the script. Future work will address optimum parallel execution on multimode clusters, and associated extension of the script.

4 CONCLUSIONS

Based on the results of this study, the parallelization of MAVRIC can be achieved in practice via the automation of the PMP approach using the P-cubed script, which has been developed for both Windows and Linux systems. The P-cubed script significantly mitigates the manual effort and

potential for errors associated with the traditional PMP method, thereby facilitating notable improvements in Monaco's performance. While the testing reported in this study was restricted to a personal computer and a workstation, the parallel efficiency data and further testing suggest that the P-cubed script enables achieving promising scalability, potentially enabling its use on larger computing clusters. It is crucial to note, however, that the performance enhancements in Monaco are expected to be dependent on both the specific problem being modelled and the hardware utilized. Future research is recommended to further explore the applicability of the P-cubed script across a broader array of problem types, as well as to investigate the relationship between Monaco performance improvements and hardware configurations.

REFERENCES

[1] E. E. LEWIS, W.F. MILLER, "Computational Methods of Neutron Transport," Wiley (1984).

[2] I. LUX, L. KOBLINGER, "Monte Carlo Particle Transport Methods: Neutron and Photon Calculations," CRC Press (1991).

[3] "SCALE: A Modular Code System for Performing Standardized Computer Analyses for Licensing Evaluation," ORNL/TM-2005/39, Version 6, Vols. I, II, and III (Jan. 2009).

[4] D. E. PEPLOW, "MAVRIC: MONACO with Automated Variance Reduction using Importance Calculations," ORNL/TM-2005039, Version 6, Vol. I, Sec. S6, Oak Ridge National Laboratory (Jan. 2009).

[5] Emmett, M. B., & Wagner, J. C. (2004). MONACO: A new 3-D Monte Carlo shielding code for SCALE. *Transactions of the American Nuclear Society*, *91*, 701-703.

[6] J. C. WAGNER, E. D. BLAKEMAN, and D. E. PEPLOW, "Forward-Weighted CADIS Method for Global Variance Reduction," *Trans. Am. Nucl. Soc.*, 97, 630 (2007).

[7] D. E. PEPLOW, T. M. EVANS, J. C. WAGNER, "Simultaneous Optimization of Tallies in Difficult Shielding Problems," *Nucl. Technol.*, 168, 785-792 (2009).

[8] Evans, T. M., Stafford, A. S., Slaybaugh, R. N., & Clarno, K. T. (2010). Denovo: A new three-dimensional parallel discrete ordinates code in SCALE. *Nuclear technology*, *171*(2), 171-200.

[9] W. A. Wieselquist, R. A. Lefebvre, Eds., SCALE 6.3.2 User Manual, ORNL/TM-2024/3386, UT-Battelle, Oak Ridge National Laboratory (February 2024).

[10] Isloor, S. S., & Marsland, T. A. (1980). The Deadlock Problem: An Overview. *Computer*, *13*(9), 58-78.