Temasek Defence Systems Institute

Temasek Defence Systems Institute

A multi-threaded cryptographic pseudorandom number generator test suite

Author : Zhang Zhibin
Thesis advisor : Mark Gondree
Second Reader : Park C. Clark

There have been recent high-profile cases of weakness in pseudorandom number generator (PRNG) leading to cryptographic flaws, resulting in security risks. One example is a flaw with OpenSSL's entropy pool on Debian, reducing its strength from 256 bits to 15 bits. It resulted in generating only 32,767 possible SSH keys of a given type and size, allowing brute force to be a practical attack on the key

For military systems where PRNG design and evaluation cannot be conducted openly, testing PRNGs empirically may be the only option available for assessing the properties of these security-critical components; this is true for the many embedded and proprietary systems employed in national security applications.

PRNG test suites provide insight and metrics for these security-critical system components. However, to date there have been no good performance analyses of PRNG test suites. Review of existing statistical test suites showed that they employed a battery of efficiently implemented tests, utilizing heavy performance optimization, but these tests were run in serial.

This thesis added multi-threading to *Dieharder*, a well-known PRNG test suite, to significantly speed up PRNG testing on multi-core systems. To implement multi-threading, two different libraries for threading were evaluated. The first implementation used a POSIX thread pool library implemented by Mark Gondree, and the second implementation used OpenMP.

Results showed that both multi-threading libraries performed better than the original *Dieharder,* with OpenMP showing shorter run-times. The run-times for OpenMP scheduling policies can be seen in Figure 1 and Figure 2. A hybrid scheduling policy was proposed to pre-sort the statistical tests before using static scheduling to evenly spread the load without incurring significant overheads.

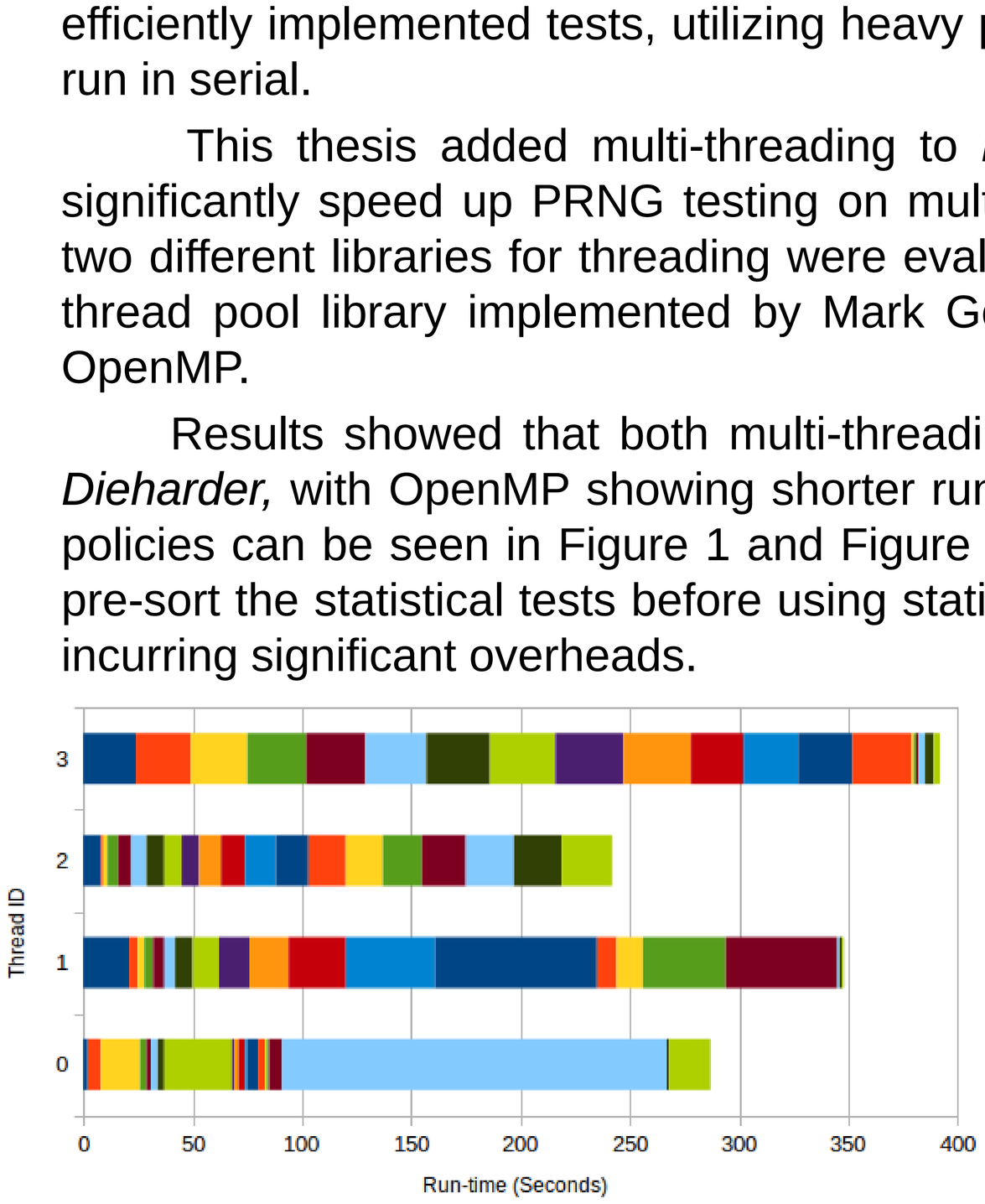

Figure 1: 4-thread Run-time for Static Scheduling

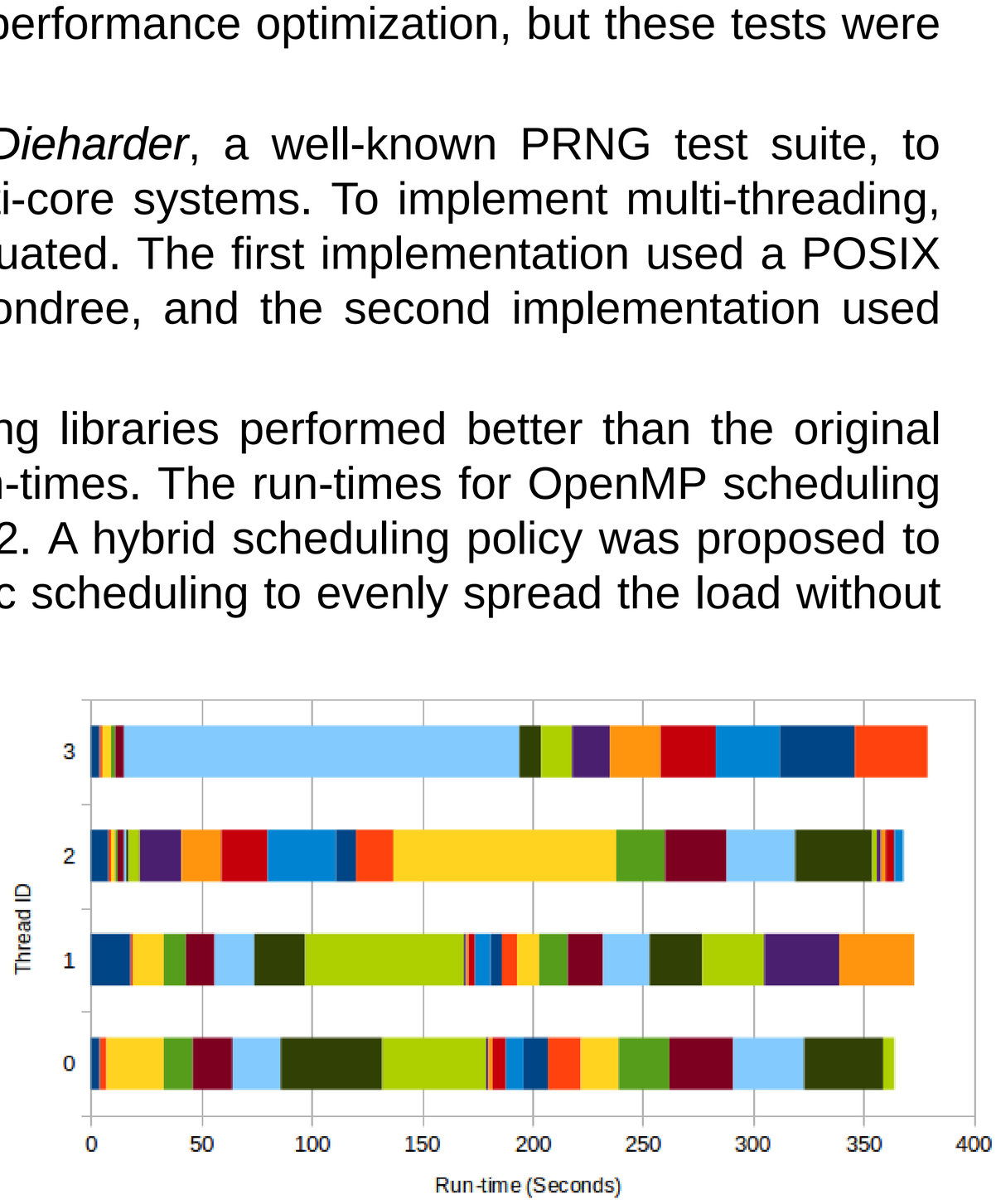

Figure 2: 4-thread Run-time for Dynamic Scheduling

NUS National University of Singapore