Parallel Pseudorandom Number Generators for Large Parallel Computations

Raj Boppana, Ph.D.
Professor and Interim Chair
Computer Science Department
University of Texas at San Antonio
Terminology

- RN: pseudorandom number
- RN stream: a sequence of RNs
- Cycle: the maximum number of RNs a stream gives before repeating the starting sequence of RNs

- RNG: pseudorandom number generator
- PRNG: parallel pseudorandom number generator
  - Provides multiple distinct RN streams

- Correlation: how predictable a set of RNs is given another set of RNs
  - Interstream and intrastream correlations
Mobile Ad Hoc Network Simulator
// node i, i varies 0 ... n-1
While (1) {
    ...
    Delta_X = RNstream(i) * MAX_DELTA;
    Delta_Y = RNstream(i) * MAX_DELTA;
    Move node to (Current_X+Delta_X, Current_Y+Delta_Y)
    ...
    // Received a broadcast packet. Retransmit it selectively
    Process the packet
    if (RNstream(i) < ReTX_Probability) {
        Jitter = RNstream(i) * MAX_JITTER;
        Broadcast Message at Current_Time + Jitter
    }
    ...
}

For better randomness, the RNs used for Jitter, node movement, re-broadcast decision, ... should be from different RN streams — tedious to manage at the application level
Results Obtained

- **Context-aware parallel random number generator (CPRNG)** [Patent pending]
  - Virtually unlimited number of random number streams, each stream with a large cycle
  - Provides distinct streams based on the location of calls for RNs (user selectable; no app. recoding)
    - Reduces the impact of intrastream correlations
  - Dynamically allocates additional RN streams beyond any upper limit specified
    - Some large, complex simulations require unpredictable and very large number of RN streams for subtasks
Results Obtained

- **Interstream Correlation (ISC) Test** [Pat. pending]
  - Evaluates correlations among parallel streams and gives a single number as quality metric
    - The first statistical test of its kind
    - The current statistical tests are 15-150 single-stream tests for intrastream correlations
  - No single quality metric; a vector of pass/fail data
  - Interstream correlations: interleave streams & apply s-s tests

- Integrates with CPRNG to provide the quality metric continually as the application consumes RNs
  - Not feasible with the current test batteries
Outline

- Related work
- Context-aware parallel pseudorandom number generator
- Interstream correlation test
- Summary and further work
Sequential RNGs

- Linear congruential generator (LCG) \( x_n = ax_{n-1} + b \pmod{2^{64}} \), \( n > 1 \)
  - Specify a, b, and the initial value \( x_0 \)
  - Parallel streams are obtained by choosing different values for a
  - 48-bit versions in Unix systems: drand48()

- Recursion with carry (RWC):
  \( c_n \parallel x_n = a_{-1}x_{n-1} + \cdots + a_{-5}x_{n-5} + c_{n-1} \pmod{2^{32}} \)
  - G. Marsaglia, Diehard package, FSU
  - RWC is used to seed the parallel RNGs we implemented

- Mersenne Twister (MT):
  \( x_{k+n} = x_{k+m} + (x_k^{\text{upper}} \parallel x_{k+1}^{\text{lower}})A, \ n > m \)
  - A is a 64x64 bit matrix
  - \( x_{k+n} \) is further multiplied by a bit matrix \( T \) to improve randomness
  - Very popular; GPU version, MTGP, by Nvidia
  - Matsumoto & Nishimura [ACM TOMACS 1998]
Parameterized Parallel RNGs

- **ALFG**
  \[ x_n = x_{n-k} + x_{n-l} \pmod{2^{64}}, \quad 0 < k < l < n \]
  - Additive lagged Fibonacci generator
  - Initialization requires specification of \(x_0, \ldots, x_{l-1}\)
  - \(2^{63}(l-1)\) distinct RN streams
  - Each with a cycle of length \(2^{63}(2^{l-1})\)
  - Lag \(l=1279\) or larger is used to minimize intra- and inter-stream correlations
  - A canonical form to initialize ALFG streams is known

- **MLFG**
  \[ x_n = x_{n-k} \ast x_{n-l} \pmod{2^{64}}, \quad 0 < k < l < n \]
  - Multiplicative lagged Fibonacci generator
  - \(2^{61}16 = 2^{976}\) distinct RN streams with lag \(l=17\)
  - Each with a cycle of length \(2^{61}(2^{17-1}) \approx 2^{78}\)
  - Lag \(l=17\) is sufficient to get high quality RN streams
Newer RNGs

- Cryptography-based RNGs
  - Use the scrambling techniques from cryptography to generate random numbers from simple sequences

- GPU friendly RNGs
  - Nvidia adapted MT for GPU implementation and offered as part of CUDA library
  - Parameterized RNGs
SPRNG: Parallel RNG and Test Package


- Provides
  - 6 PRNGs: ALFG, MLFG, two LCGs and two other generators
  - Several single stream tests
    - Interleaves multiple parallel RN streams to form a sequential stream and applies sequential tests for interstream correlation checks
  - Ising model (in Physics) simulation codes using Metropolis and Wolff algorithms
    - Exact solutions are known so that the simulation error can be estimated

- Used for initial implementation of CPRNG
Test Packages

- Statistical tests to analyze intrastream correlations of a single stream
  - Dieharder (Brown, Duke University)
  - TU01 (‘Ecuyer & Blouin, ACM TOMS, 2007)

- Application-based testing
  - 2-D Ising model simulations
    - Exact theoretical results known
    - Simulations are used to evaluate the quality of RNGs
    - Two algorithms for simulations: Metropolis and Wolff
Context-aware parallel pseudorandom number generator (CPRNG)

Based on the Fibonacci recurrence:

$$x_n = x_{n-k} \times x_{n-l} \pmod{2^{64}}, \ 0 < k < l < n$$

Implementation: an include file + a small library module
CPRNG Initialization of RN Streams

Random Fill using RWC and user-specified seed Common to all RN streams

Distinct ID
RN Context
RN Context
CPRNG Initialization: RN Context

- Calls for RNs with the same stream ID, but from different locations, result in the use of distinct RN streams.

- Random number context consists of:
  - Stream identifier: 0, ..., total_str-1
  - Program location: return address of the function call to RNG, or program line number and file name
  - Optional: process ID, thread ID, iteration number

Lag L-1
Lag L-2
Lag K+2
Lag K+1
Lag K
Lag 0
MLFG is part of SPRNG
CPRNG1 implemented in Phase I inside SPRNG package
CPRNG2 implemented in Phase II
CPRNG1 takes 1ns more than MLFG because of API compatibility issues.

CPRNG2 does not have the same issues.
Testing the randomness of parallel pseudorandom number generators
**Single-stream Tests**

- Collisions
- Coupon collector’s
- Equidistribution
- Gap
- Maximum-of-t
- Permutations
- Poker
- Runs up
- Serial
- Random walk
- Matrix rank test

**Test packages with 15 to 150 single-stream tests**
- **Dieharder, TestU01, SPRNG**
- Empirical in nature
- Vector of pass/fail data: hard to compare different RNGs

**Statistical Test applied k times**

- \( p\text{-value}_1 \in [0,1] \)
- \( p\text{-value}_2 \)
- \( p\text{-value}_k \)

**Kolmogorov-Smirnov (KS) Test**

- Pass/Fail
Current Parallel RN-Stream Tests

- Single-stream tests evaluate intrastream correlations in a stream.

- Interstream correlations of parallel RN streams are evaluated by:
  - Interleaving the RN streams into a single stream
  - Applying single-stream tests

- Alternatively, simulations of system/process models with known exact solutions may be used:
  - 2-D Ising model in Physics: Metropolis and Wolff algorithms
Interstream Correlation (ISC) Test

RN Streams

Mixer

\( \rho \)

Mixer

\( \rho \)

Mixer

\( \rho \)

Correlation Coefficient

Aggregator

(K-S or D-R test)

PRNG Quality Metric
Application of ISC Test

- CPRNG and MLFG were tested
- Up to 1.5 billion RN streams used
  - 1500 sets of RN streams, 10 to 1 million per set
  - 100+ RNs from each stream
  - Shuffle interleaving of streams in a set
- Even the largest test case completes in 6 hours on a desktop with i7-870 CPU, 12 GB mem
If the absolute value of the DR-statistic is above 1.96, then the correlations among the RN streams are considered significant. The probability of false conclusion is 0.05.

A set of $10$ to $10^6$ RN streams gives one cor. coeff., $r$

1500 $r$'s are used to calculate the statistic
If the absolute value of the KS-statistic is above 0.0274, then the correlations among the RN streams are considered significant. The probability of false conclusion is 0.01.
Adapting ISC Test for Online Testing

PRNG

Application

Results

ISC Test

PRNG
Quality Metric

Test Specification

Test Specification

Open Cloud Laboratory
Silicon Informatics
Summary and Further Work

- CPRNG provides two new features that other generators do not provide
  - Distinct RN streams based on program context
  - Virtually unlimited number of RN streams without requiring the user to specify a maximum upper bound

- CPRNG is implemented with simple API
  - An API definition file and a library module
  - Wrappers can be built to adapt CPRNG for the applications that use SPRNG or default RNGs on various platforms
  - Seamless integration of CPU and GPU modes
    - The same streams can be used in both modes
Summary and Further Work

- ISC test: a new statistical test to evaluate interstream correlations of billions of RN streams
  - Can be adapted to provide a quality metric continually while the application runs and after it completes

- Ongoing work
  - Intel Xeon Phi implementation of CPRNG
  - Refine GPU implementation of CPRNG
  - Looking for HPC users
    - Use cases for CPRNG and ISC test
    - HPC application testing of ISC test and CPRNG context-awareness feature
    - Visit getcprng.com and register to obtain additional information
Acknowledgments

- This work was supported by STTR Phase I & II grants from ARO (Program manager: Dr. Joe Myers)

- The desktop computer used for development was acquired with funds from an NSF grant; additional computer resources were provided by UTSA’s Institute for Cyber Security

- STTR project participants
  - Prof. Ravi Sandhu and Prof. Ram Tripathi, UT San Antonio
  - Mr. Robert Keller and Mr. Hemant Trivedi, Silicon Informatics
  - Prof. Srinivasan, Florida State U.

This presentation does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

Raj Boppana
boppana@cs.utsa.edu
210-458-4434