



# ROOT and MIXMAX

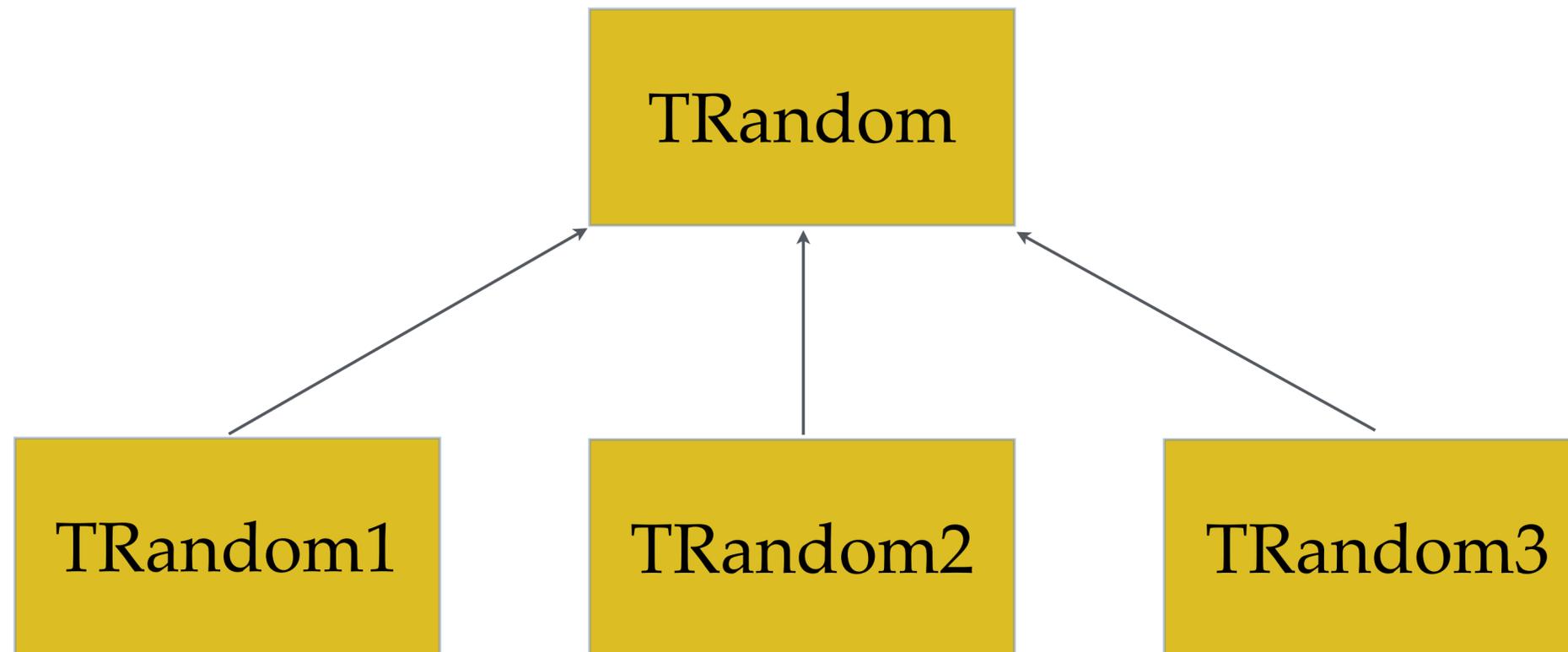
L. Moneta (CERN)

# Outline

- Random Numbers in ROOT
- Integration of MIXMAX generator in ROOT
- Performance of random generators in ROOT
- Conclusions

# ROOT Random Number Classes

- In ROOT the pseudo-random numbers are generated using the TRandom classes



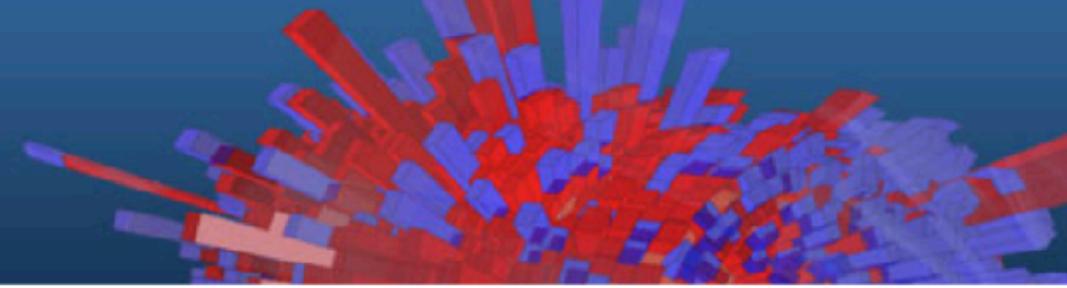
Random number generators before introducing MIXMAX

# TRandom

- TRandom is the base class implementing common functions such as
  - `TRandom::Uniform`
  - `TRandom::Gaus`
  - `TRandom::Poisson`
  - `TRandom::Exp`
- `TRandom::Rndm` is the virtual function generating a PRNG double number in ]0,1]
  - internal generator is a Linear Congruential Generator (LCG).  
It is kept for backward compatibility, but it should not be used
    - $\mathbf{x}_{n+1} = (\mathbf{a}x_n + \mathbf{c}) \% \mathbf{m}$  with  $\mathbf{a}=1103515245$ ,  $\mathbf{c}=12345$  and  $\mathbf{m}=2^{31}$

```
// generate a number in interval ]0,1]
double number = gRandom->Rndm();
// generate a gaussian number
double x = gRandom->Gauss(mean, sigma);
// generate a Poisson number with <n>=mu
double n = gRandom->Poisson(mu);
```

# New Random Number Design



- New design for Random number classes has been introduced recently in ROOT
- Clear separation between
  - engine classes (that generates the numbers)
  - generation of numbers according to distributions
- Classes implementing algorithms for generation of pseudo-random numbers
  - e.g. `ROOT::Math::MersenneTwisterEngine`, `ROOT::Math::MixMaxEngine`
- Class for implementing most used random number distributions (e.g. Gauss, Poisson, Binomial,...)
  - `ROOT::Math::RandomFunctions`
- User interface class
  - `ROOT::Math::Random`

# MIXMAX generator in ROOT

- Since ROOT version 6.05.02 MIXMAX has been included as a new Engine class in the new design
  - `ROOT::Math::MixMaxEngine` (wrapper to the C 2.0 version)
    - `TRandomMixMax256`, the modified 1.0 version with N=256, skipping 2 iterations
    - `TRandomMixMax` (with N=240)
    - `TRandomMixMax17` (N=17) using less memory and with faster seeding
- All engines pass the TestU01 suite

# MIXMAX Engine Class

- MIXMAX pseudo-random number generator has been integrated in the `ROOT::Math::MixMaxEngine` class
  - simple wrapper to the original C implementation (from MIXMAX version 2.0)

```
template<int N, int S>
class MixMaxEngine : public TRandomEngine {

public:

    typedef TRandomEngine BaseType;

    MixMaxEngine(uint64_t seed=1);

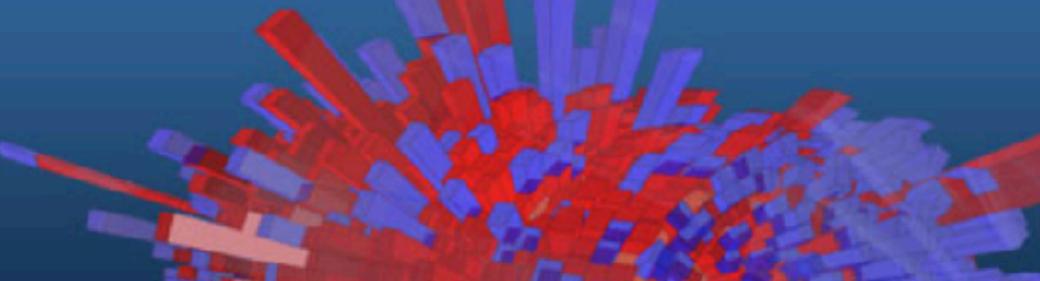
    /// Get the size of the generator
    static int Size();

    /// set the generator seed
    void SetSeed(Result_t seed);

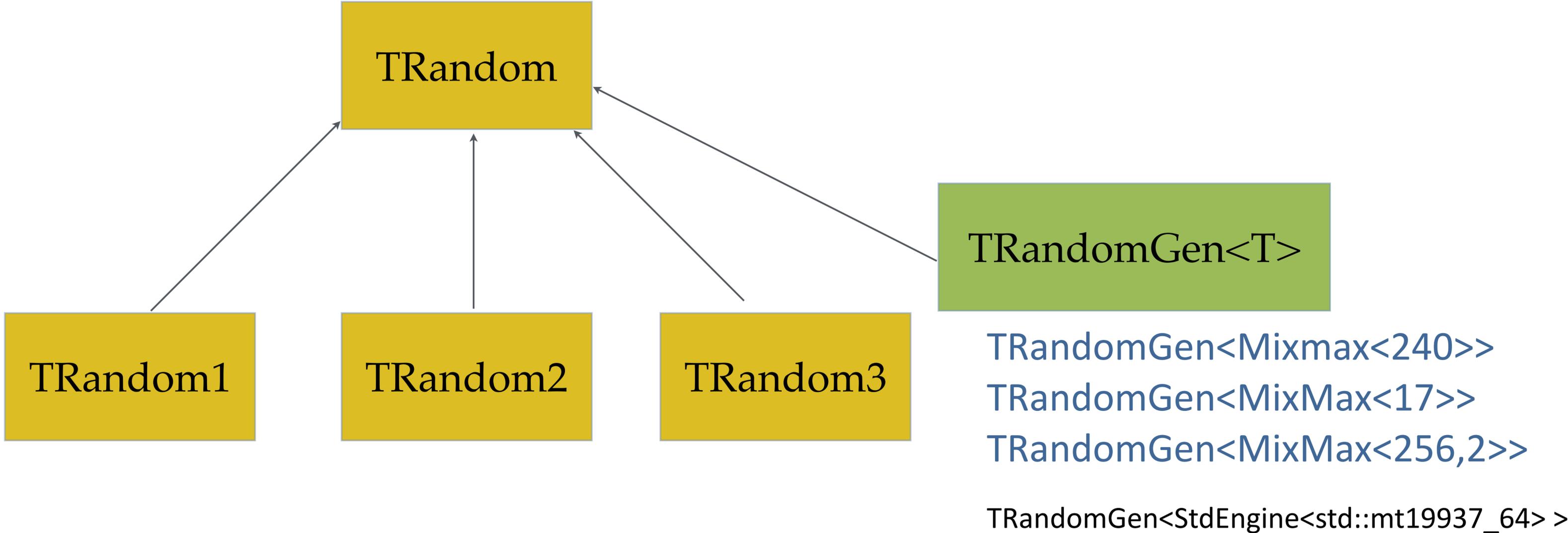
    /// generate a random number
    virtual double Rndm() { return Rndm_impl(); }

    inline double operator() () { return Rndm_impl(); }
```

# New Random Number Design



- MIXMAX integrated in ROOT TRandom classes



# Example of Usage

```
TRandom3 r(seed); // Use Mersenne-Twister generator

double number = r.Rndm(); // generate number in [0,1]

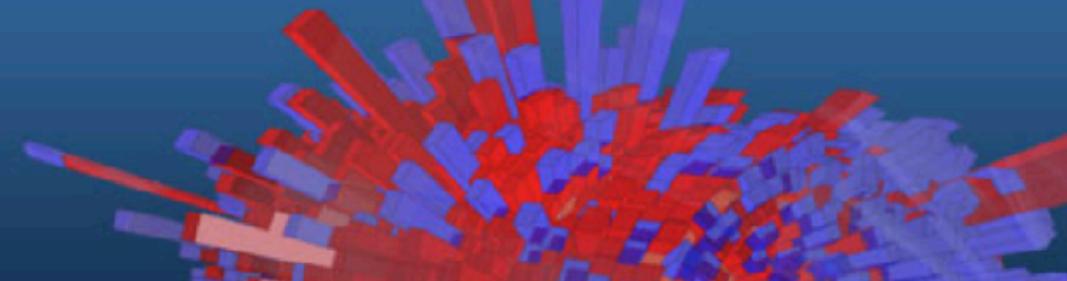
number = r.Gauss(mean,sigma); // generate a normal number

// using MIXMAX
TRandomMixMax17 r2(seed); // Use MixMax 17. TRandomMixMax17 is a
                          // typedef to TRandomGen<MixMax<17>
int n = r2.Poisson(4.2) // generate a Poisson number with mu=4.2

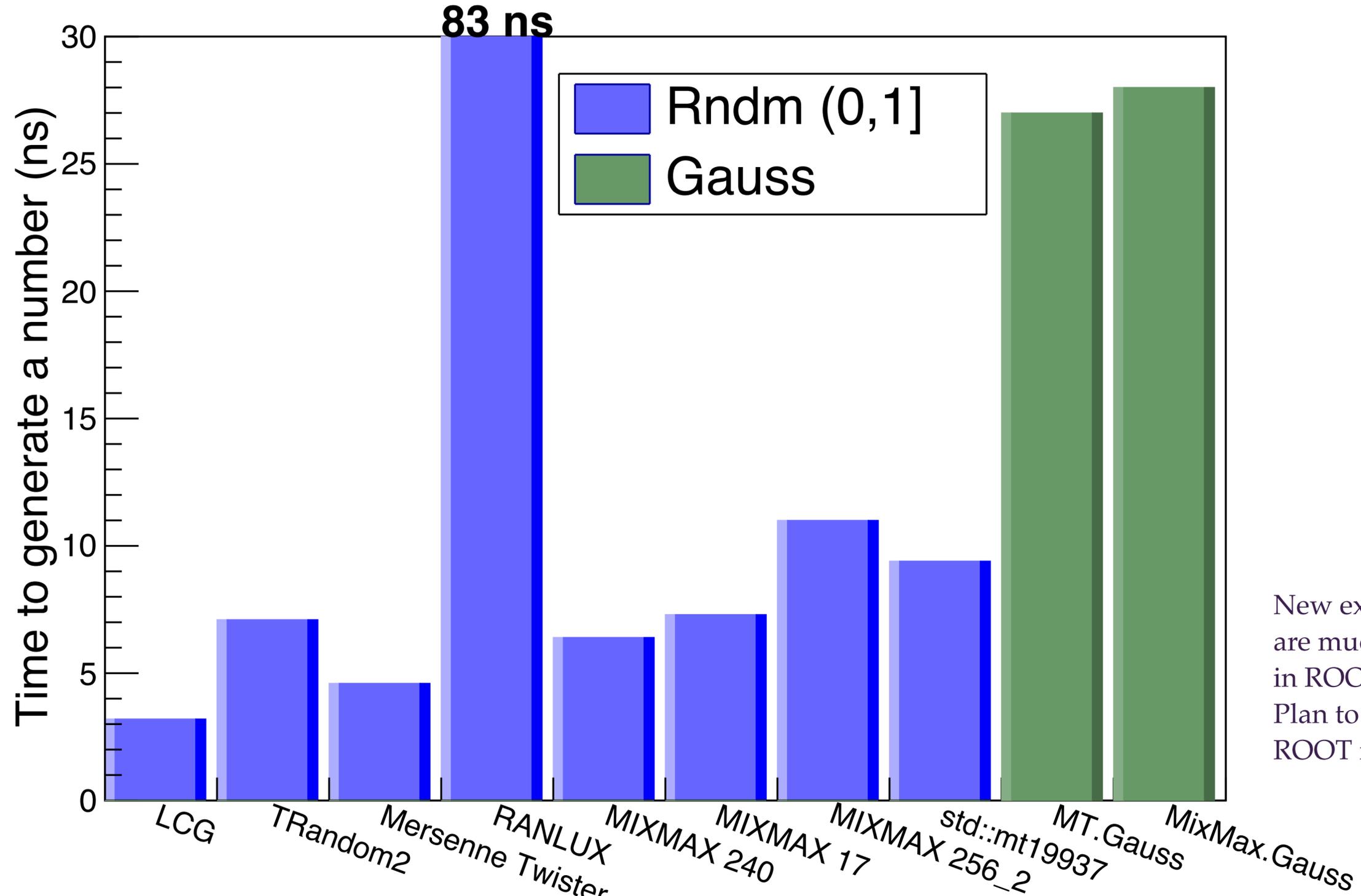
gRandom = new TRandomMixMax256(seed); // Use MixMax 256

number = histogram->GetRandom(); // generate random number from
                                  // histogram
```

# PRNG CPU Performance



CPU performance for PRNGs available in ROOT



New existing RANLUX versions are much faster than current one in ROOT (TRandom1). Plan to integrate them for next ROOT release

# Conclusions

- Random numbers are an important component in ROOT
- Uses for statistical studies (e.g bootstrapping), Monte Carlo integration
- Fast seeding is essential especially when using them in parallel applications
- Plan to integrate an improved version of RANLUX, which are much faster than current version available in ROOT
- Work on improving performances using SIMD vectorisation in random engines