```python
import numpy as np
import cupy as cp
import time

def numpy_random_matrix(dim):
    return np.random.rand(dim, dim).astype(np.float32)

def numpy_random_vector(dim):
    return np.random.rand(dim).astype(np.float32)

def cupy_random_matrix(dim):
    return cp.random.rand(dim, dim, dtype=cp.float32)

def cupy_random_vector(dim):
    return cp.random.rand(dim, dtype=cp.float32)

def numpy_dot(matrix, vector):
    return matrix.dot(vector)

def cupy_dot(matrix, vector):
    return matrix.dot(vector)

def compare_performance(dim):
    print("\nComparing performance for dimension N = {}\n".format(dim))

    # Generar matriz y vector en la CPU utilizando NumPy
    start = time.time()
    np_matrix = numpy_random_matrix(dim)
    np_vector = numpy_random_vector(dim)
    numpy_result = numpy_dot(np_matrix, np_vector)
    numpy_time = time.time() - start

    # Generar matriz y vector en la GPU utilizando CuPy
    start = time.time()
    cp_matrix = cupy_random_matrix(dim)
    cp_vector = cupy_random_vector(dim)
    cupy_result = cupy_dot(cp_matrix, cp_vector)
    cupy_time = time.time() - start

    print("NumPy (CPU) Time: {:.6f} seconds".format(numpy_time))
    print("CuPy (GPU) Time: {:.6f} seconds".format(cupy_time))

    speedup = numpy_time / cupy_time
    print("GPU is {:.2f} times faster than CPU for dimension N = {}".format ↪
      (speedup, dim))


# Test con diferentes valores de N

compare_performance(10000)
compare_performance(20000)
compare_performance(30000)
compare_performance(40000)
```