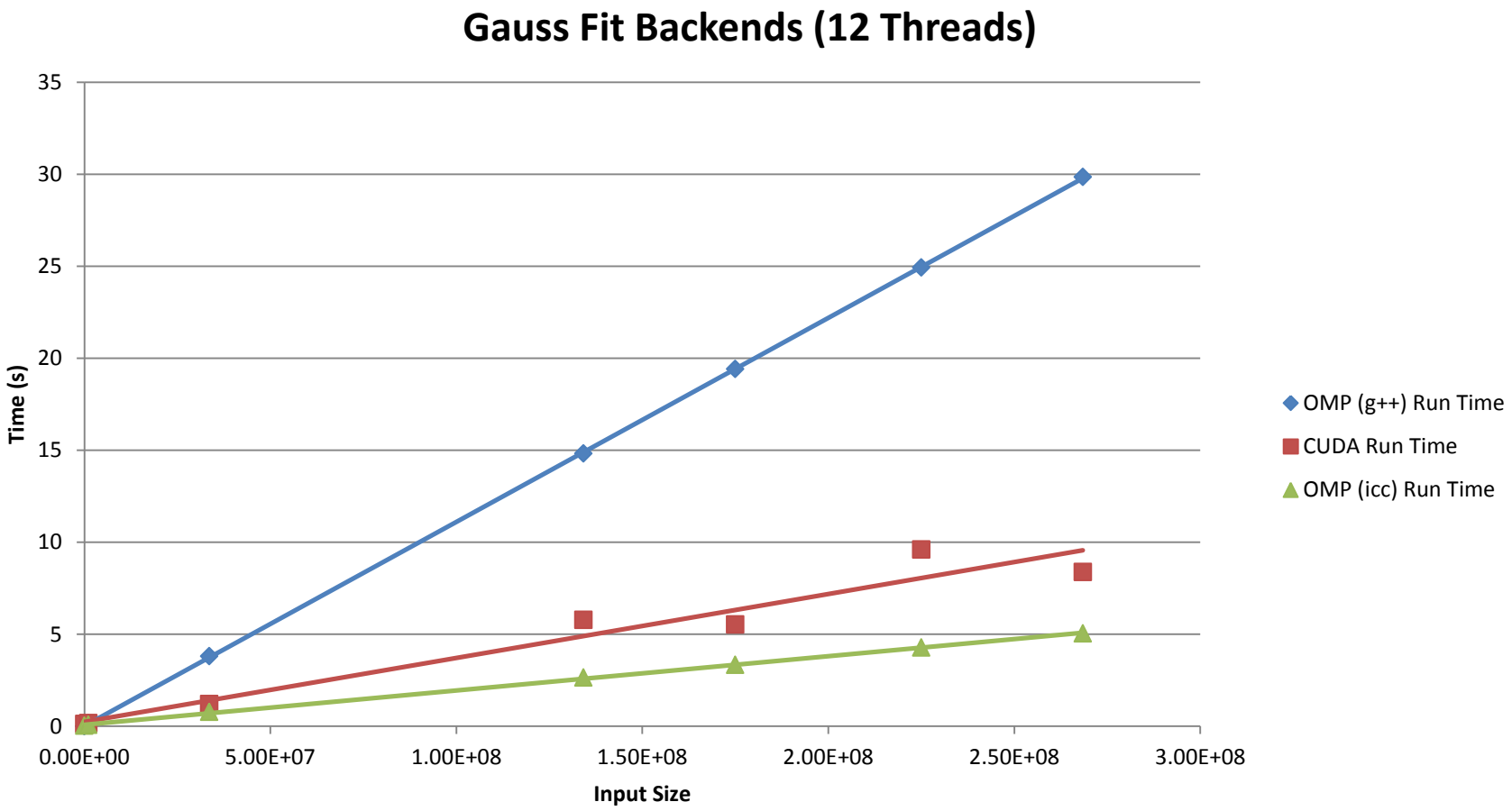
Thrust OpenMP Backend

OpenMP Backend on Intel Westmere Xeon Processors

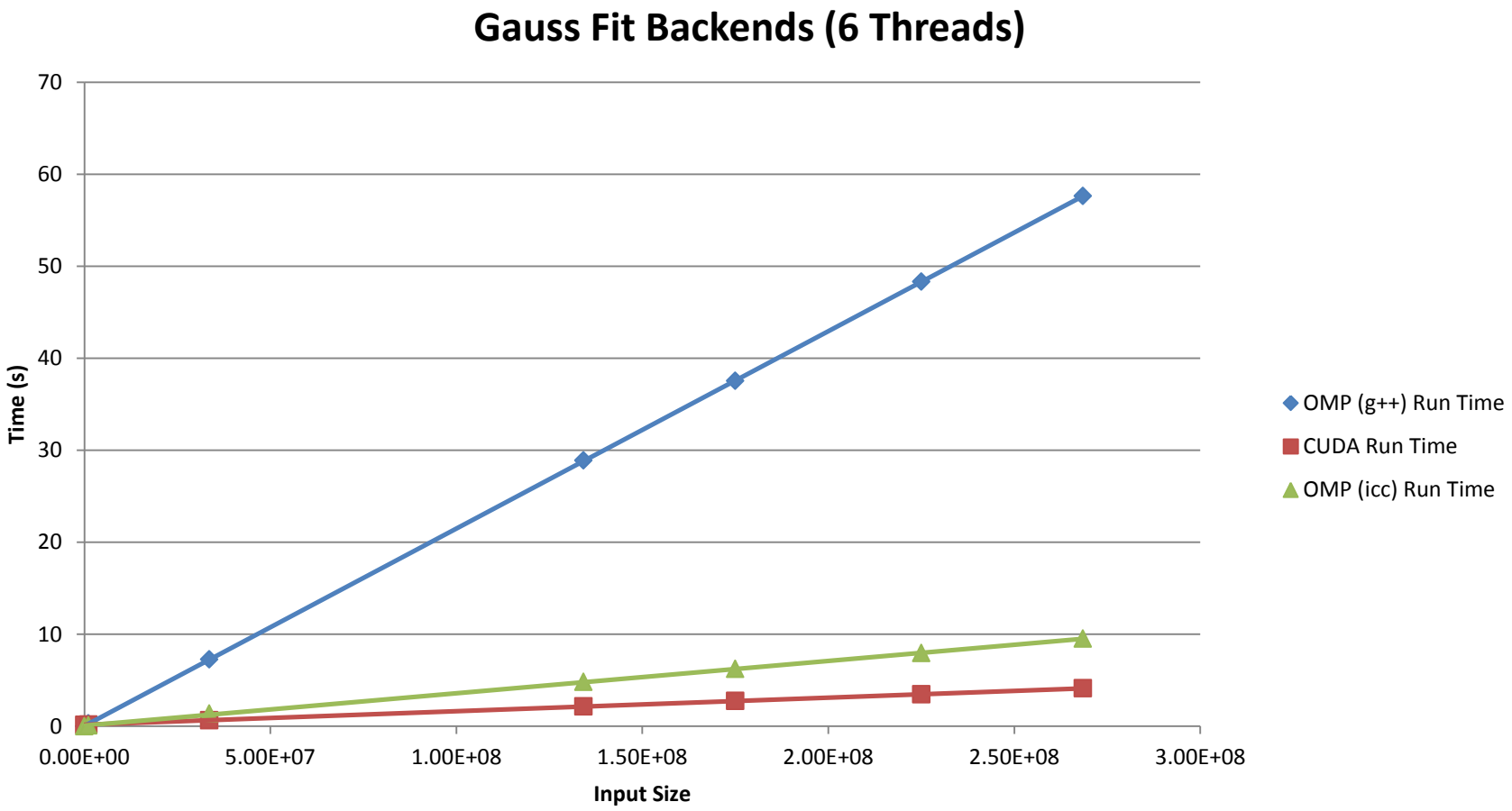
Compile (Gaussian Fit)

- Compiling with CUDA Backend
  - nvcc -o a.out -arch=sm_20 solution2.cu -L./rootstuff/ -lRootUtils
- Compiling with OpenMP Backend (icc and g++)
  - g++ -O2 -o omp.out solution2.cpp -fopenmp -DTHRUST_DEVICE_BACKEND=THRUST_DEVICE_BACKEND_OMP -lgomp -I $CUDA_HOME/include -L./rootstuff/ -lRootUtils
  - icc -O2 -o omp.out solution2.cpp -fopenmp -DTHRUST_DEVICE_BACKEND=THRUST_DEVICE_BACKEND_OMP -lgomp -I $CUDA_HOME/include -L./rootstuff/ -lRootUtils

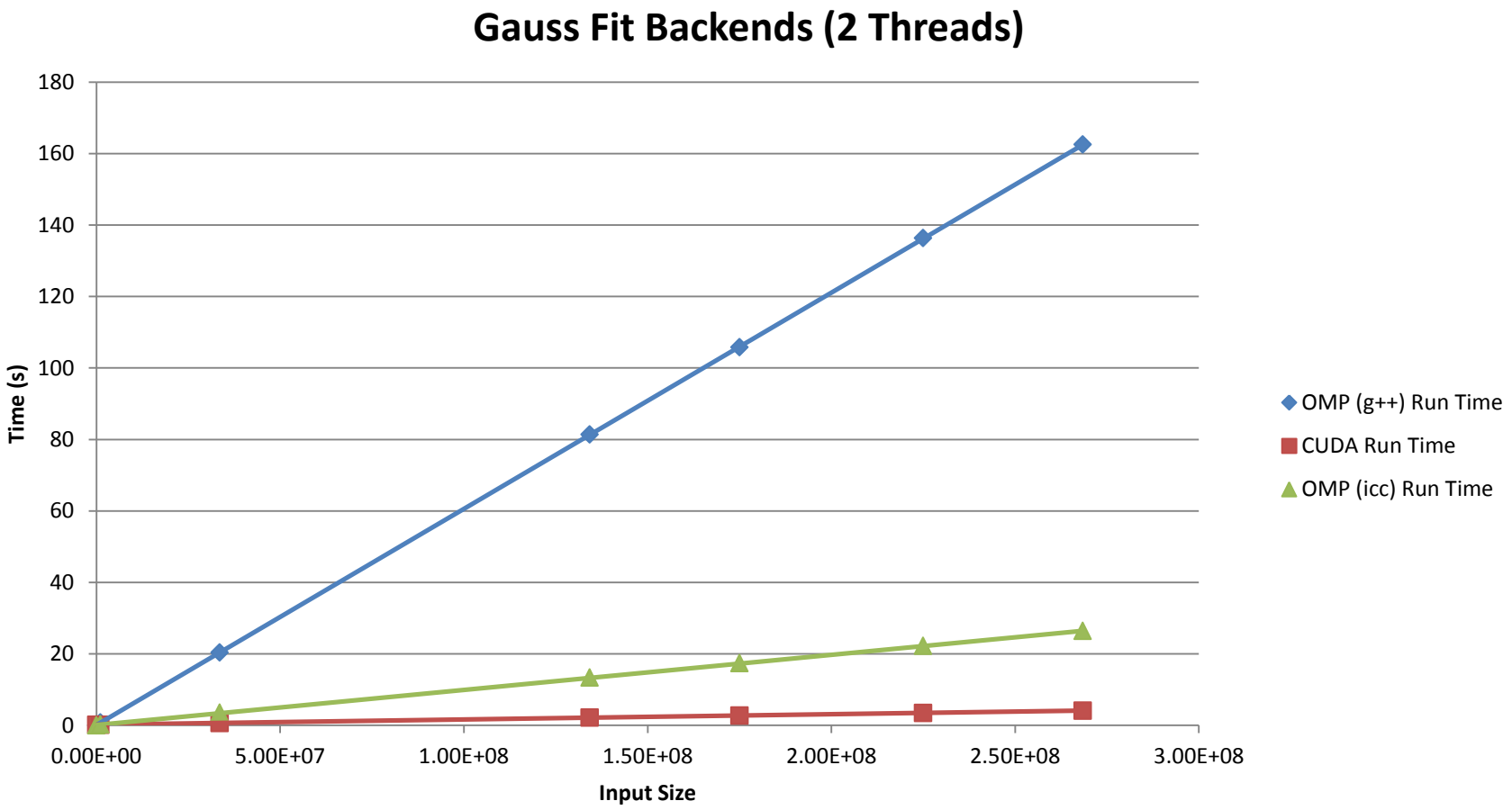
Gauss Fit: CUDA vs OpenMP (12 Threads)



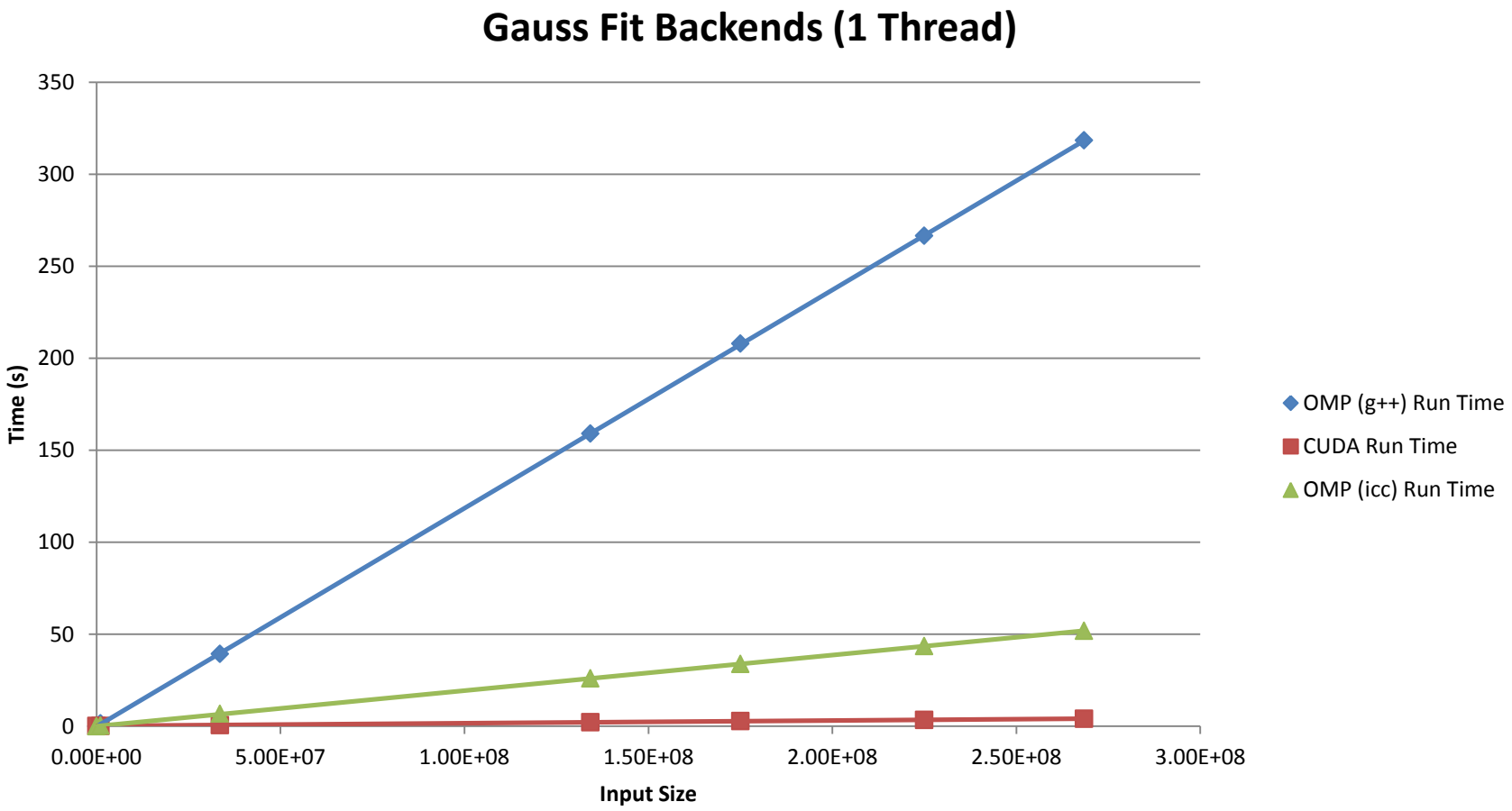
Gauss Fit: CUDA vs OpenMP (6 Threads)



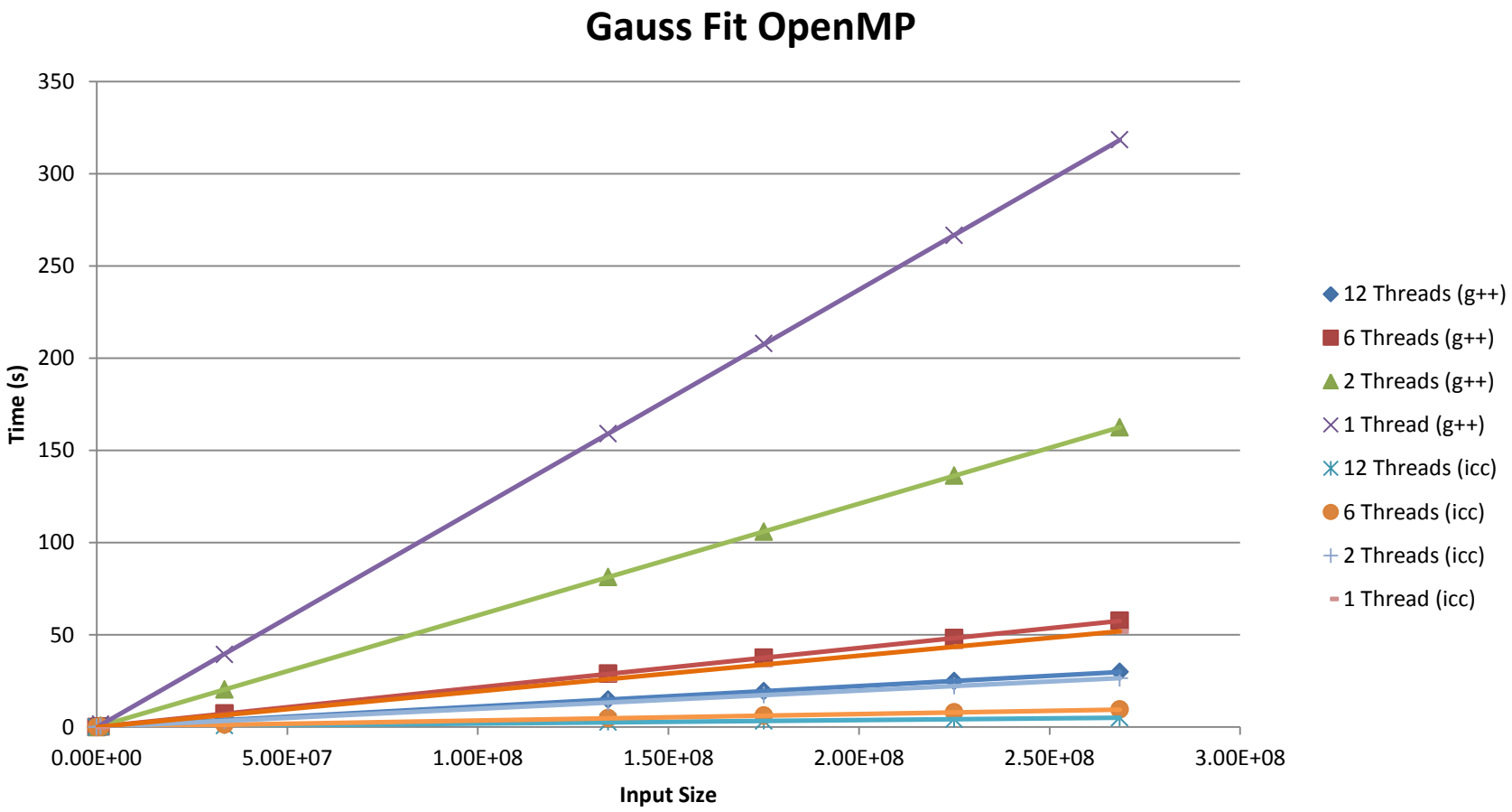
Gauss Fit: CUDA vs OpenMP (2 Threads)



Gauss Fit: CUDA vs OpenMP (1 Thread)



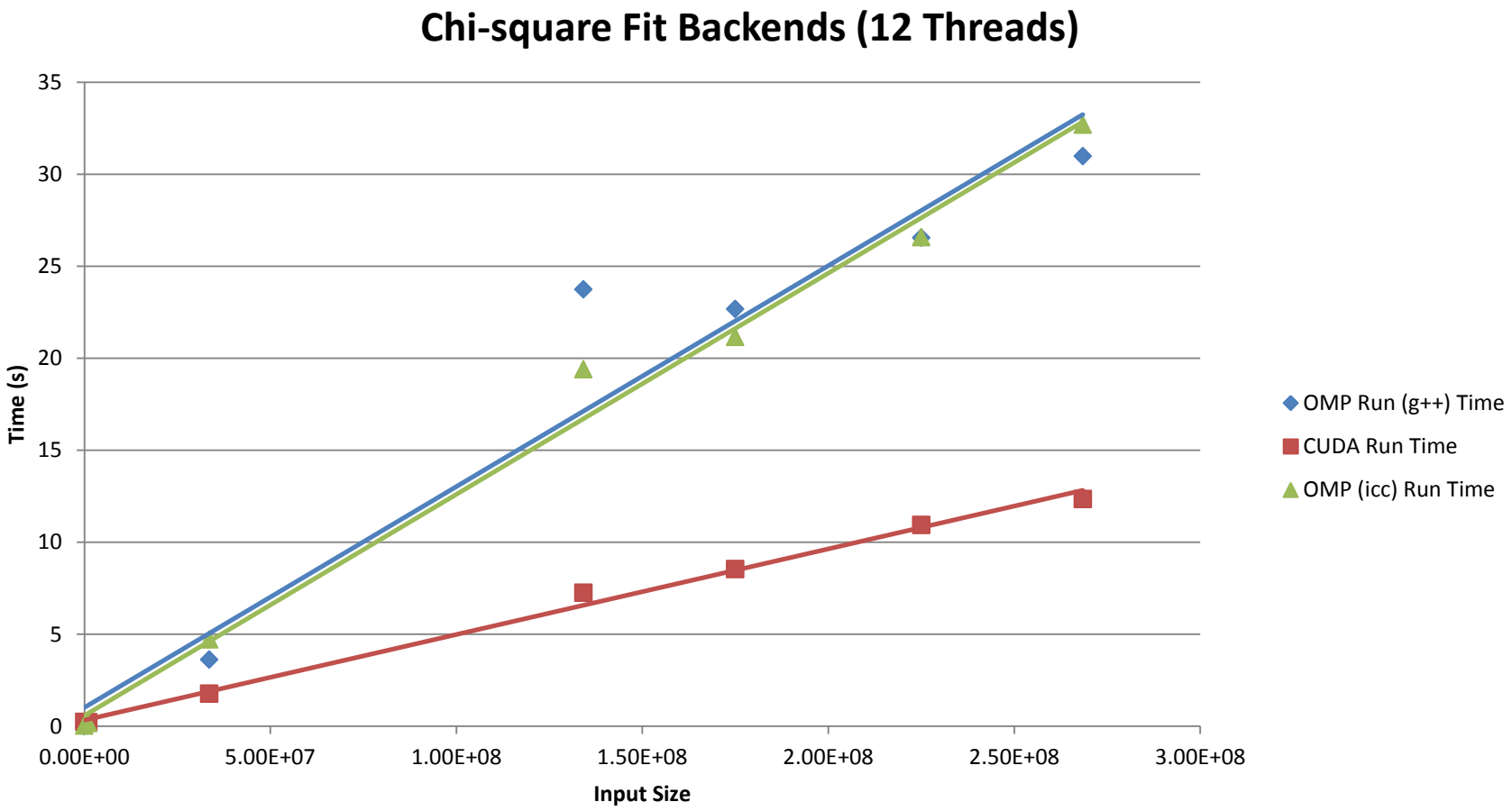
Gauss Fit: Number of OpenMP Threads



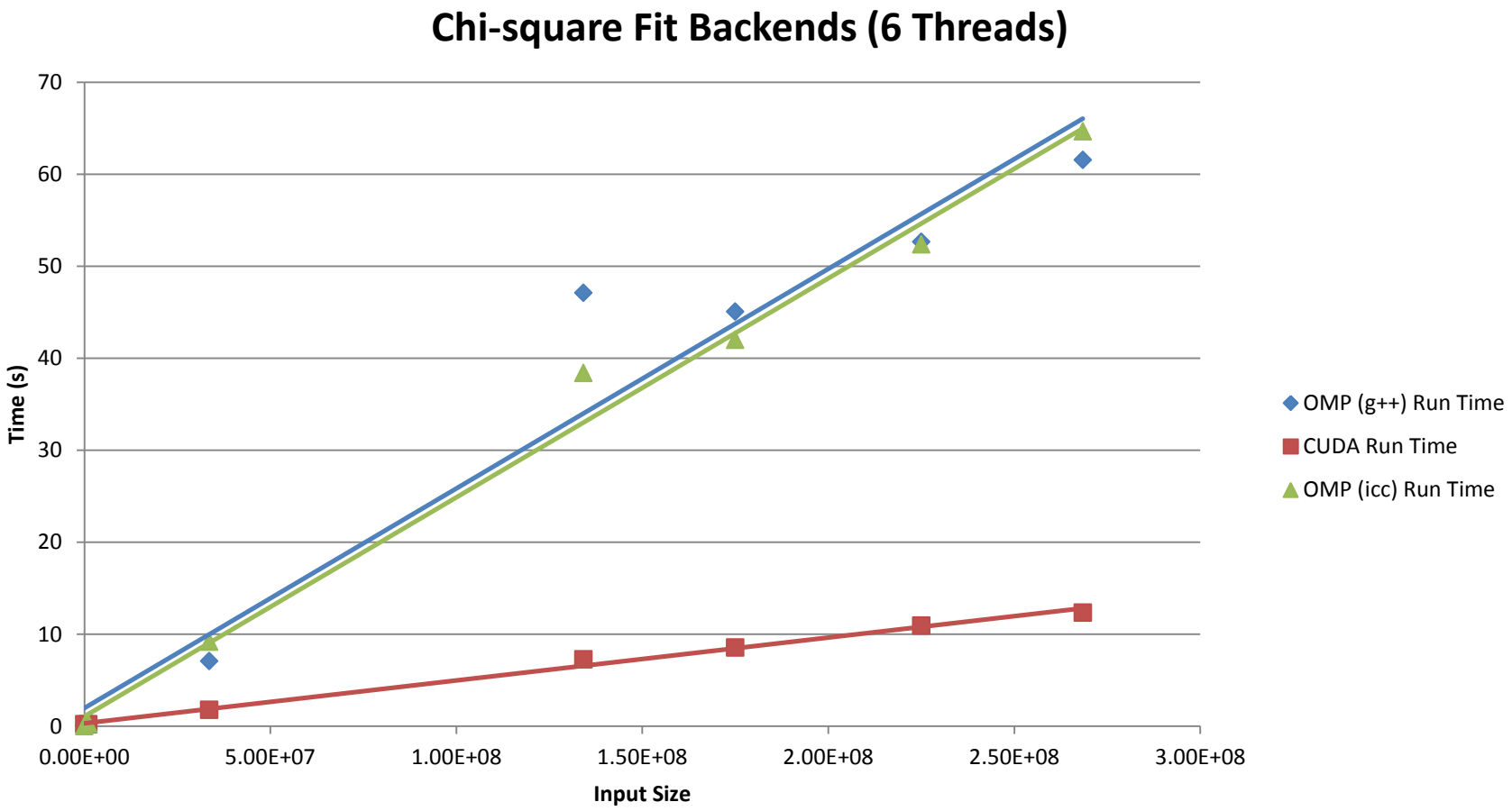
Compile (Chi-square Fit)

- Compiling with CUDA Backend
  - nvcc -o a.out -arch=sm_20 example2.cu -L./rootstuff/ -lRootUtils
- Compiling with OpenMP Backend
  - g++ -O2 -o omp.out example2.cpp -fopenmp -DTHRUST_DEVICE_BACKEND=THRUST_DEVICE_BACKEND_OMP -lgomp -I $CUDA_HOME/include -L./rootstuff/ -lRootUtils
  - icc example2.cpp -o omp.out -fopenmp -DTHRUST_DEVICE_BACKEND=THRUST_DEVICE_BACKEND_OMP -lgomp -I $CUDA_HOME/include -O2 -L./rootstuff/ -lRootUtils

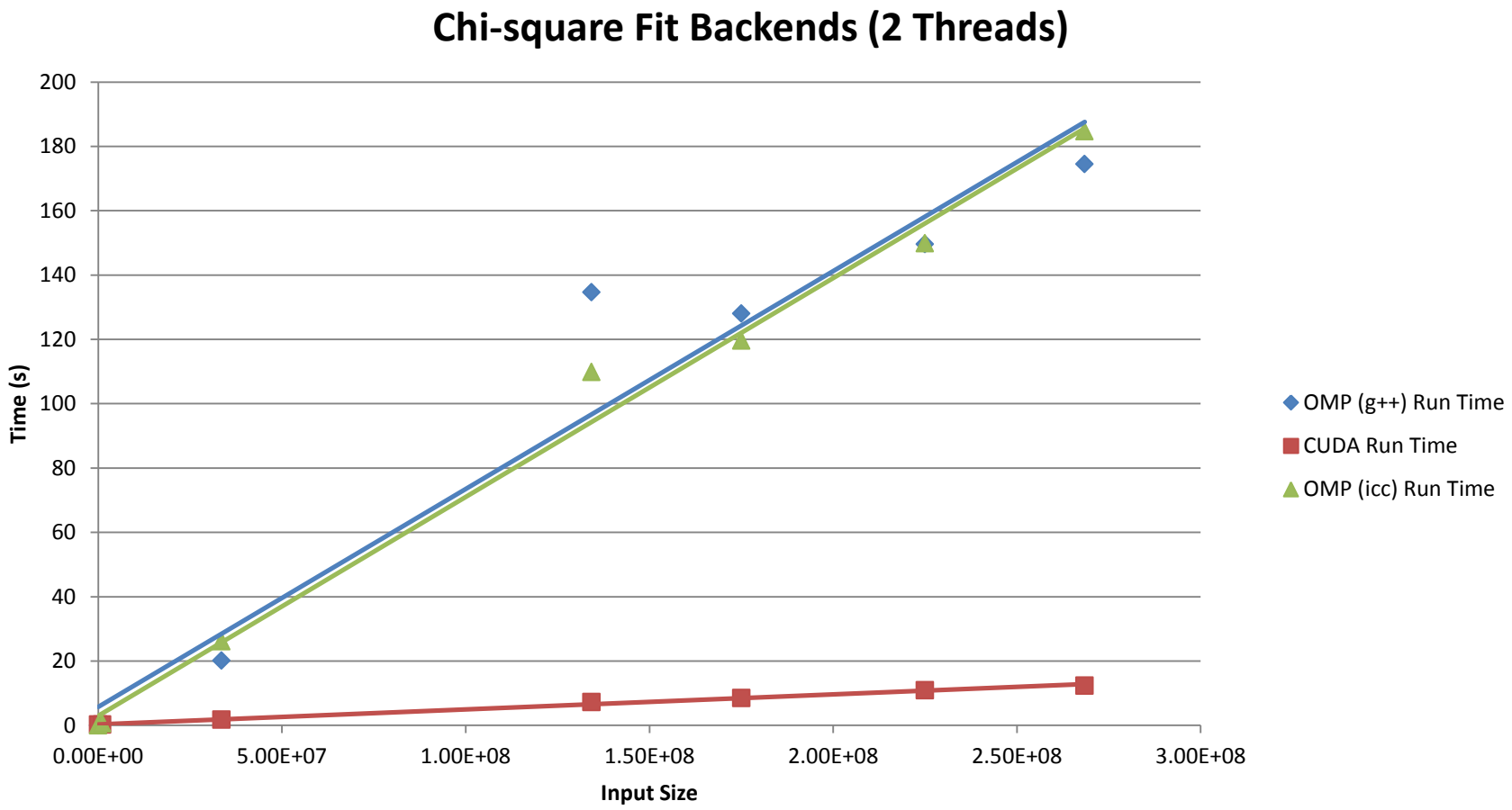
Chi-square Fit: CUDA vs OpenMP (12 Threads)



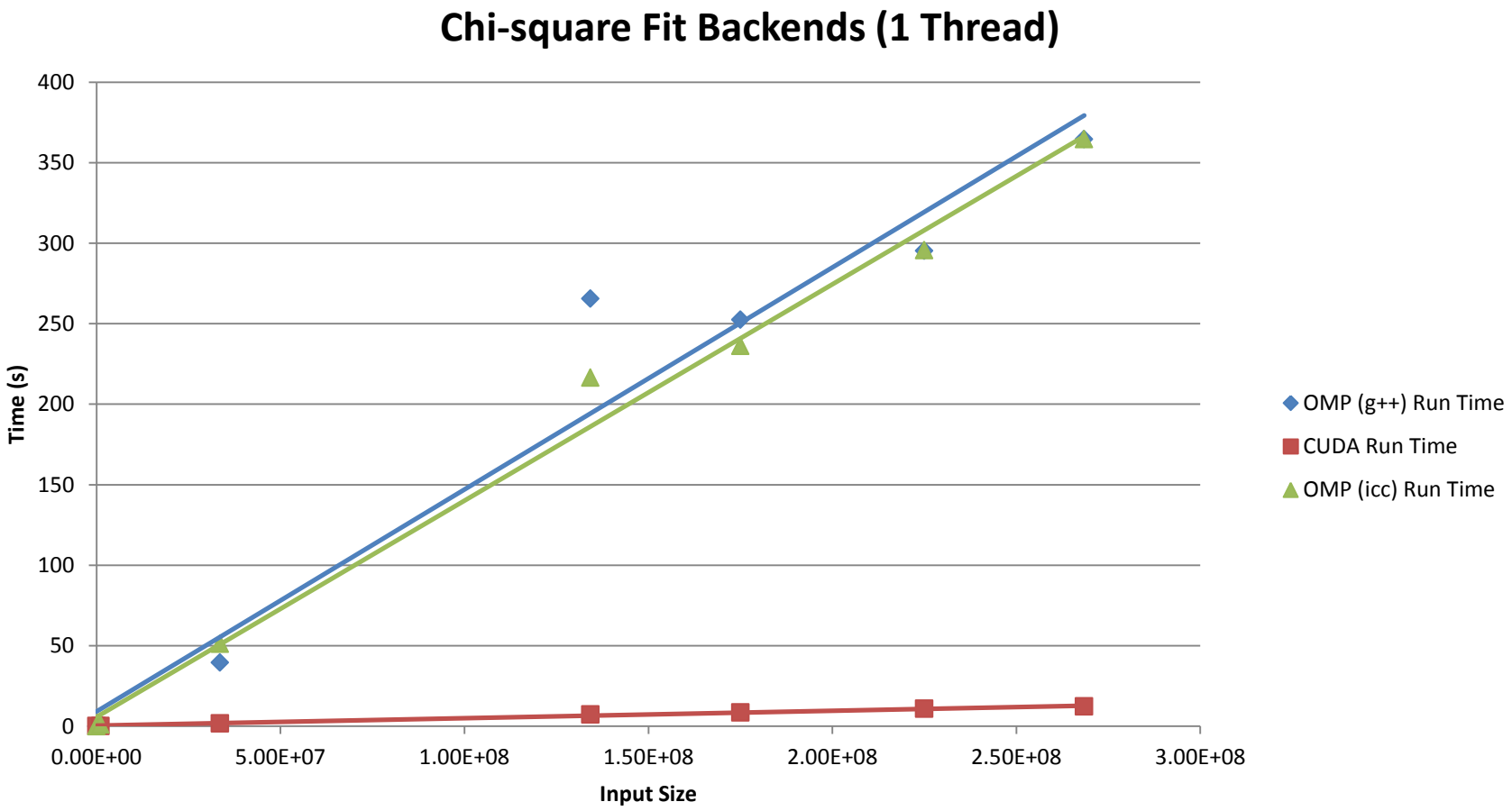
Chi-square Fit: CUDA vs OpenMP (6 Threads)



Chi-square Fit: CUDA vs OpenMP (2 Threads)

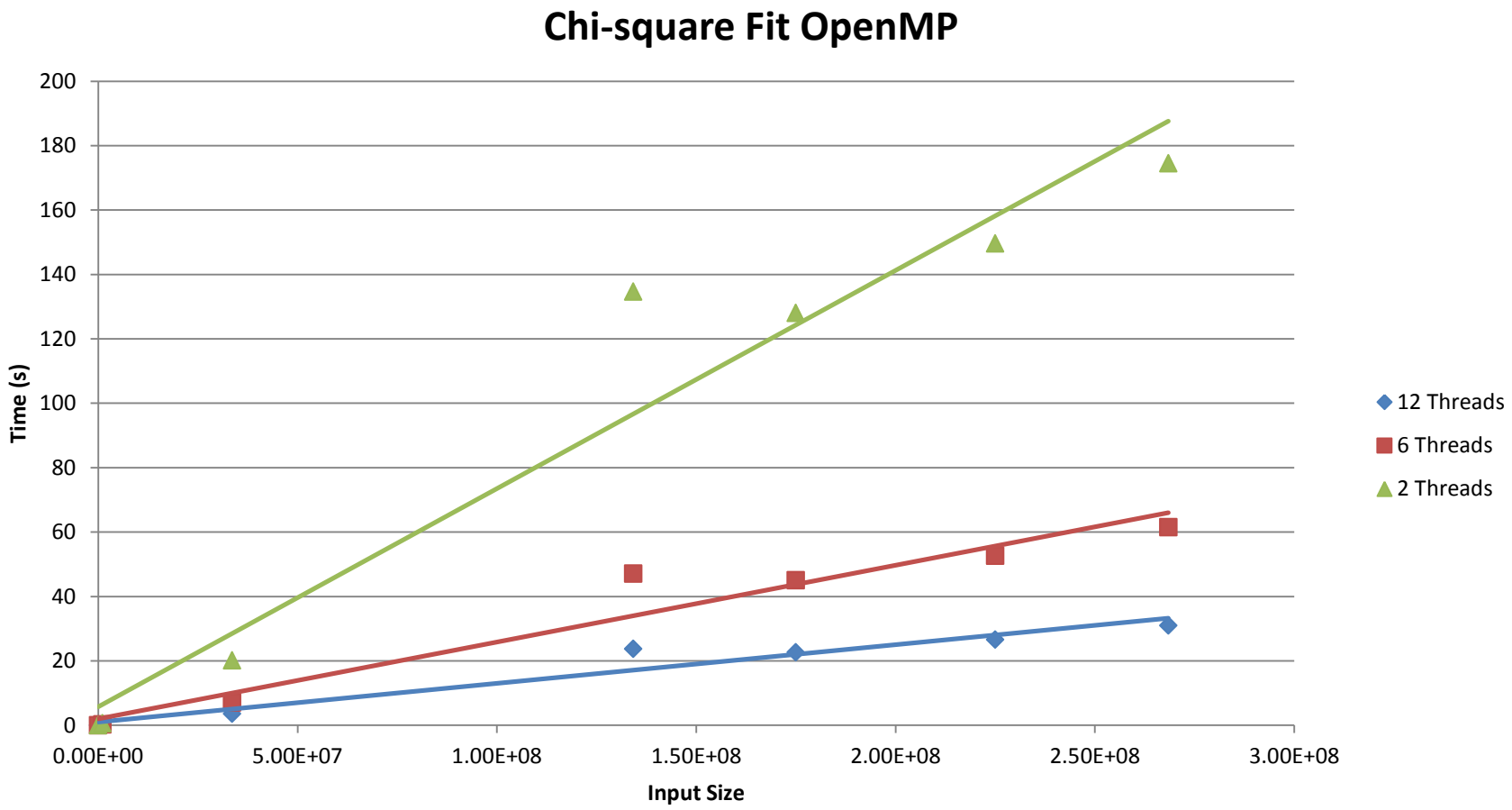


Chi-square Fit: CUDA vs OpenMP (1 Thread)



Chi-square Fit: Number of OpenMP Threads

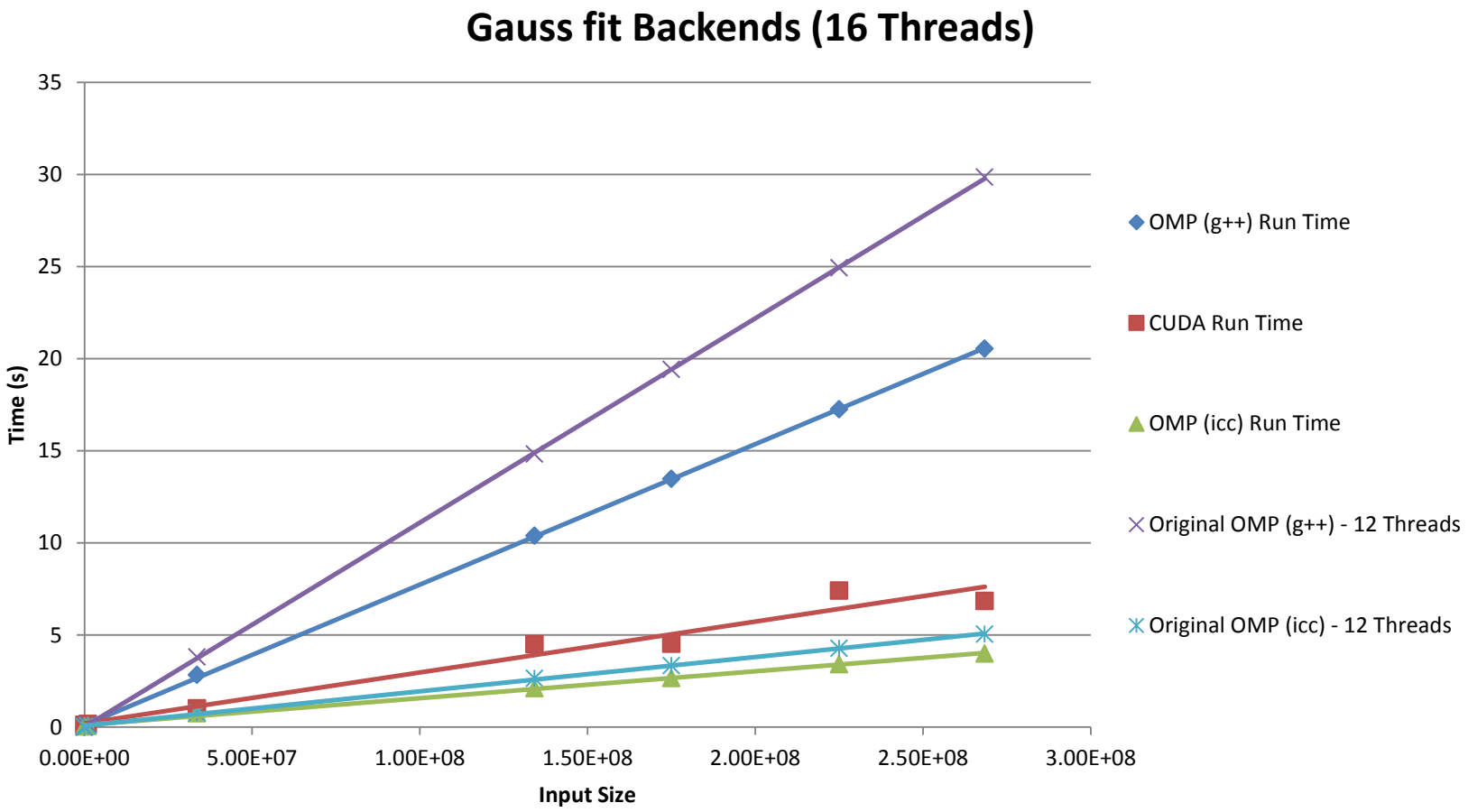
(icc not included because times were similar)



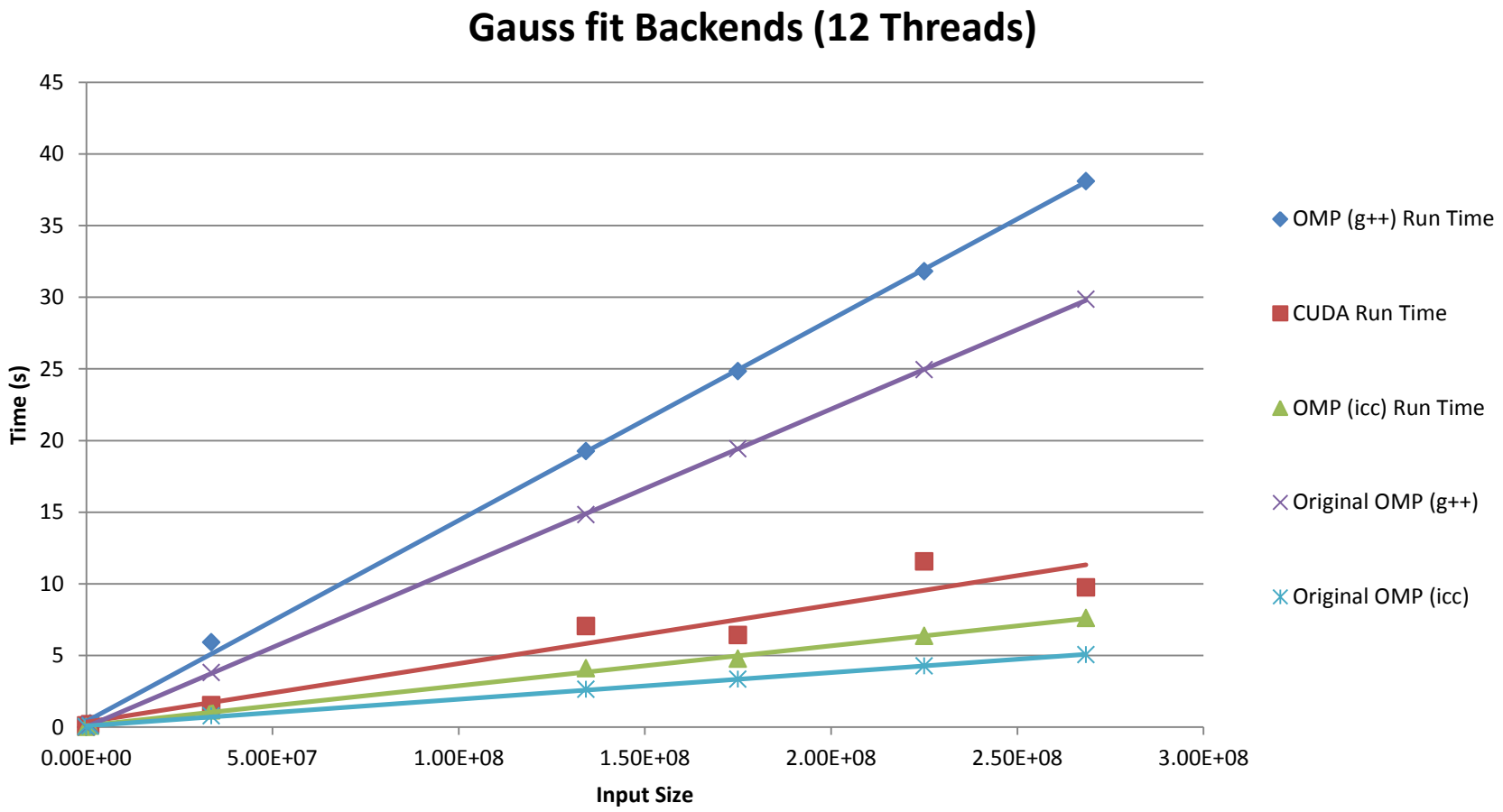
OpenMP Backend on Intel Sandybridge Xeon processors

(Chi-square fit algorithm optimized)

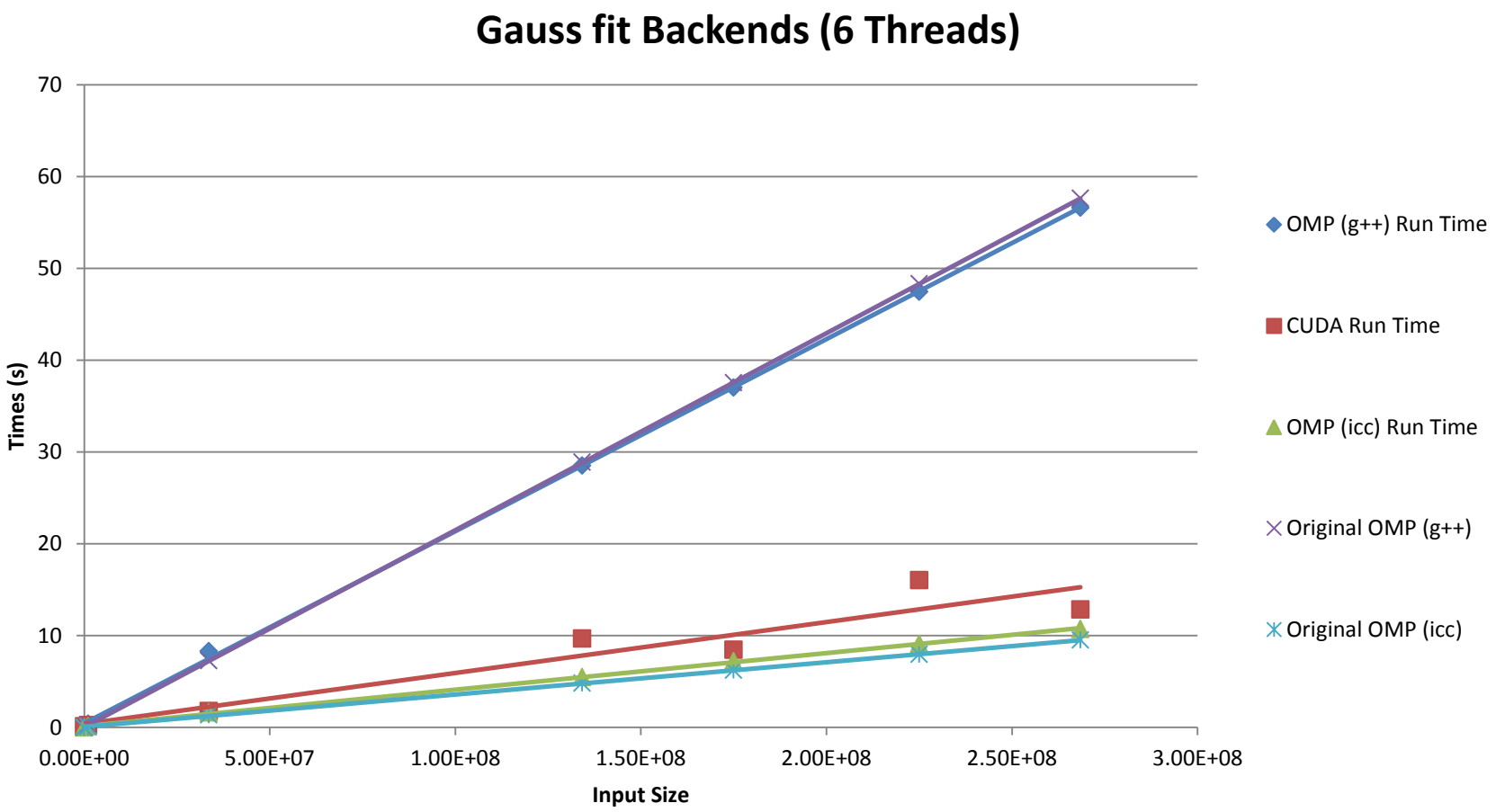
Gauss Fit: CUDA vs. OpenMP (16 Threads)



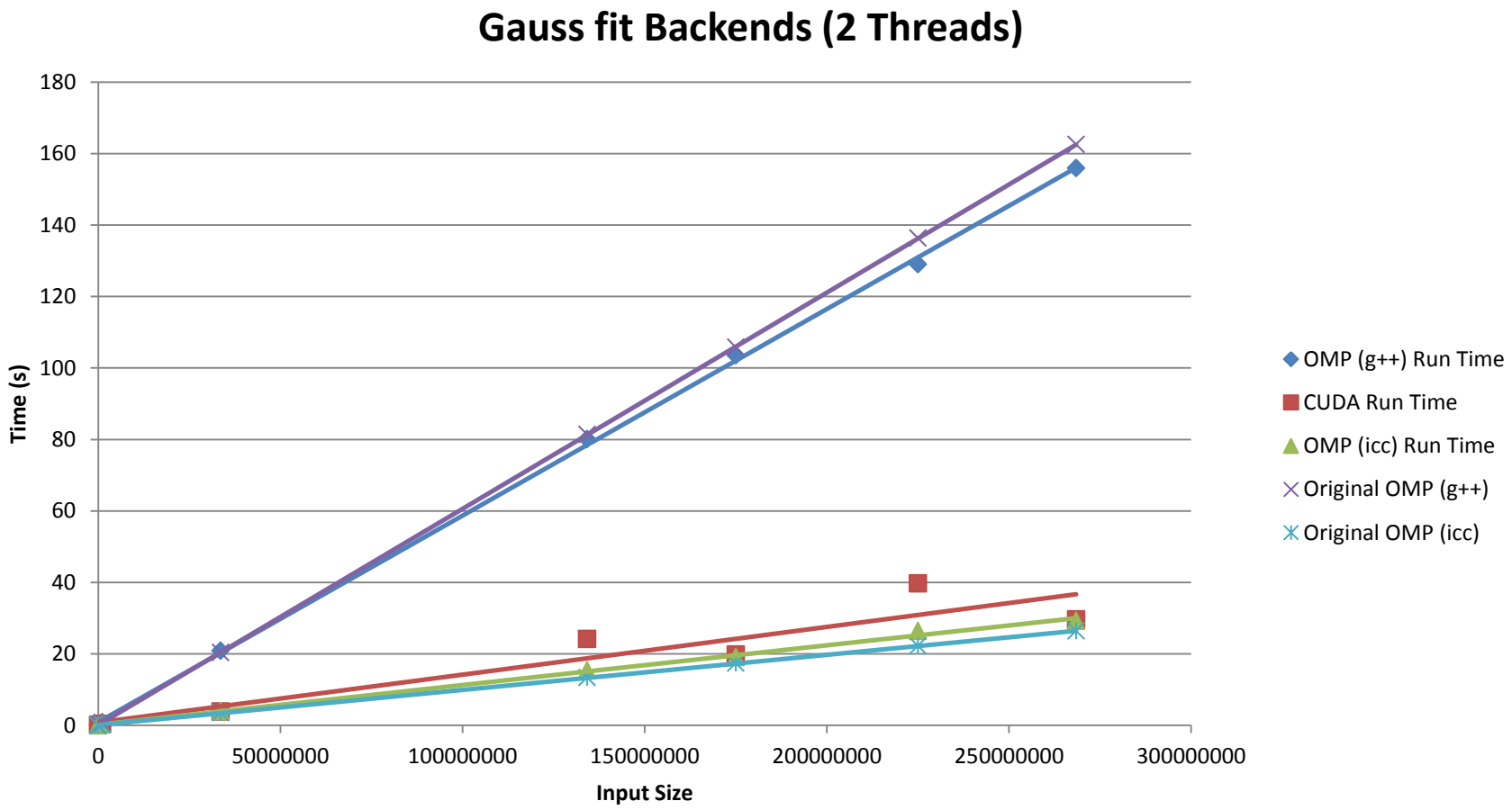
Gauss Fit: CUDA vs. OpenMP (12 Threads)



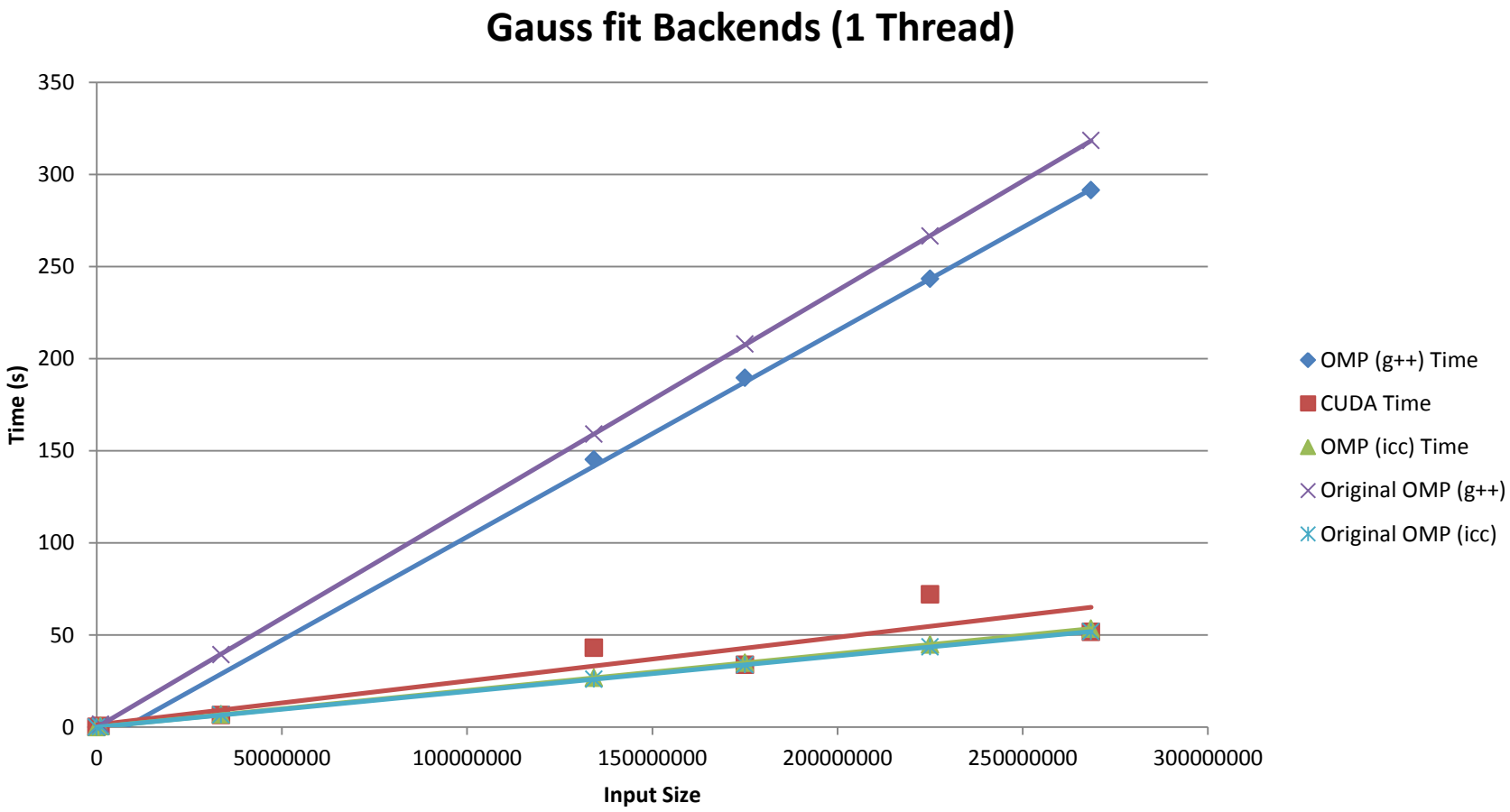
Gauss Fit: CUDA vs. OpenMP (6 Threads)



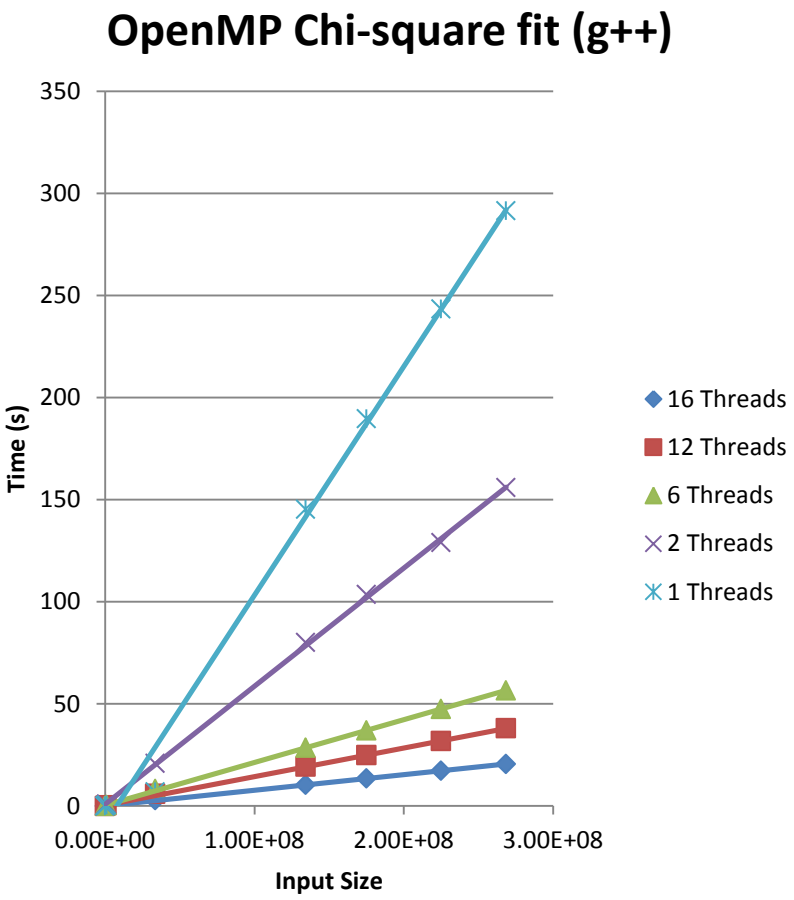
Gauss Fit: CUDA vs. OpenMP (2 Threads)

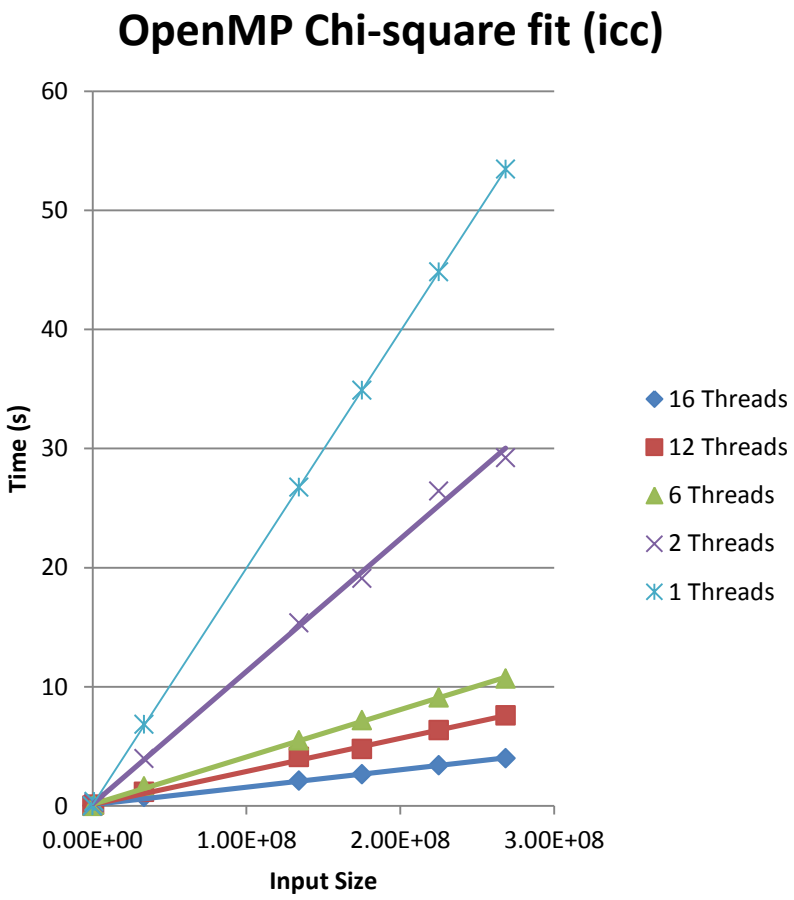


Gauss Fit: CUDA vs. OpenMP (1 Threads)



Gauss Fit: Number of OpenMP Threads

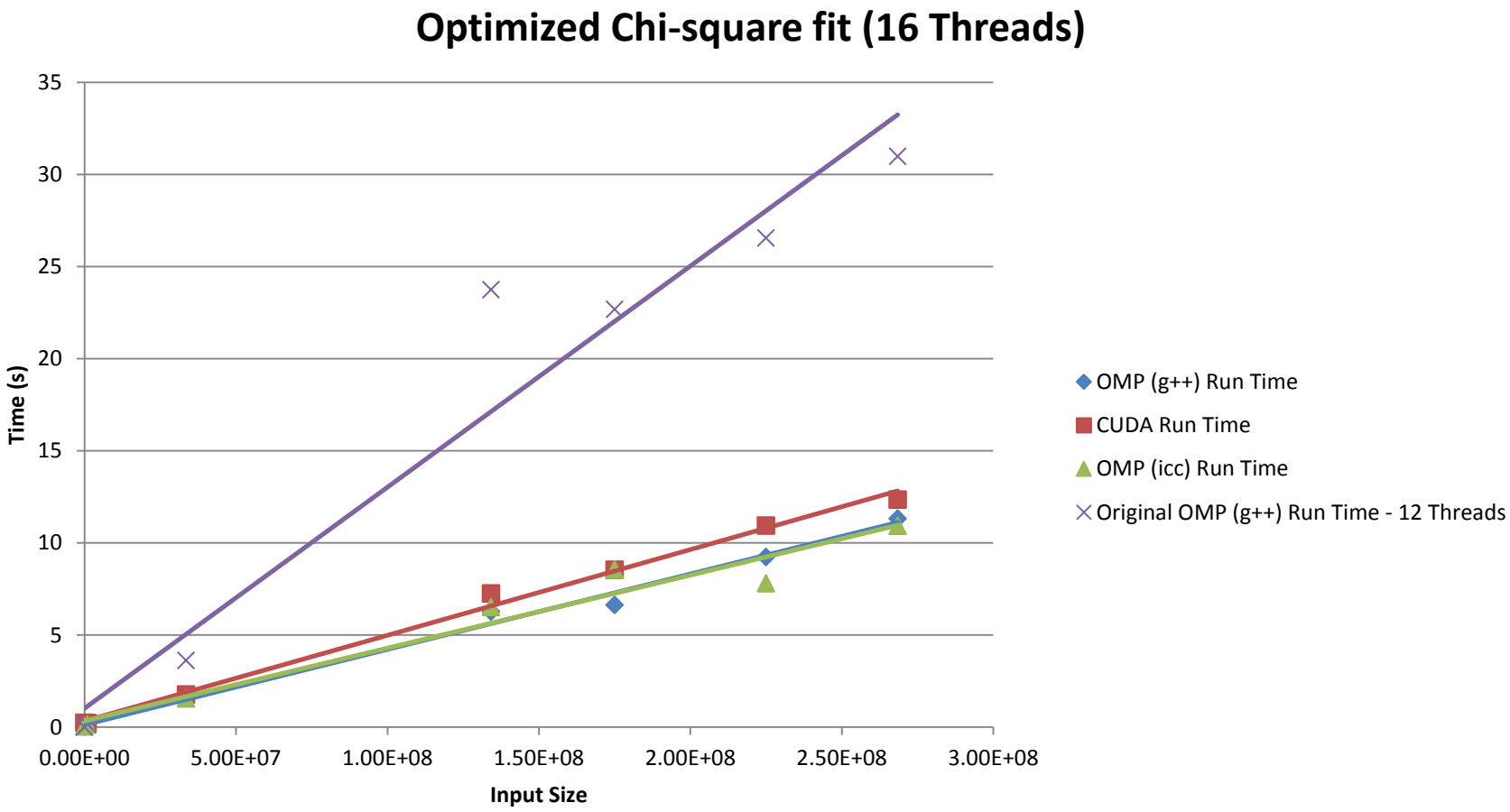




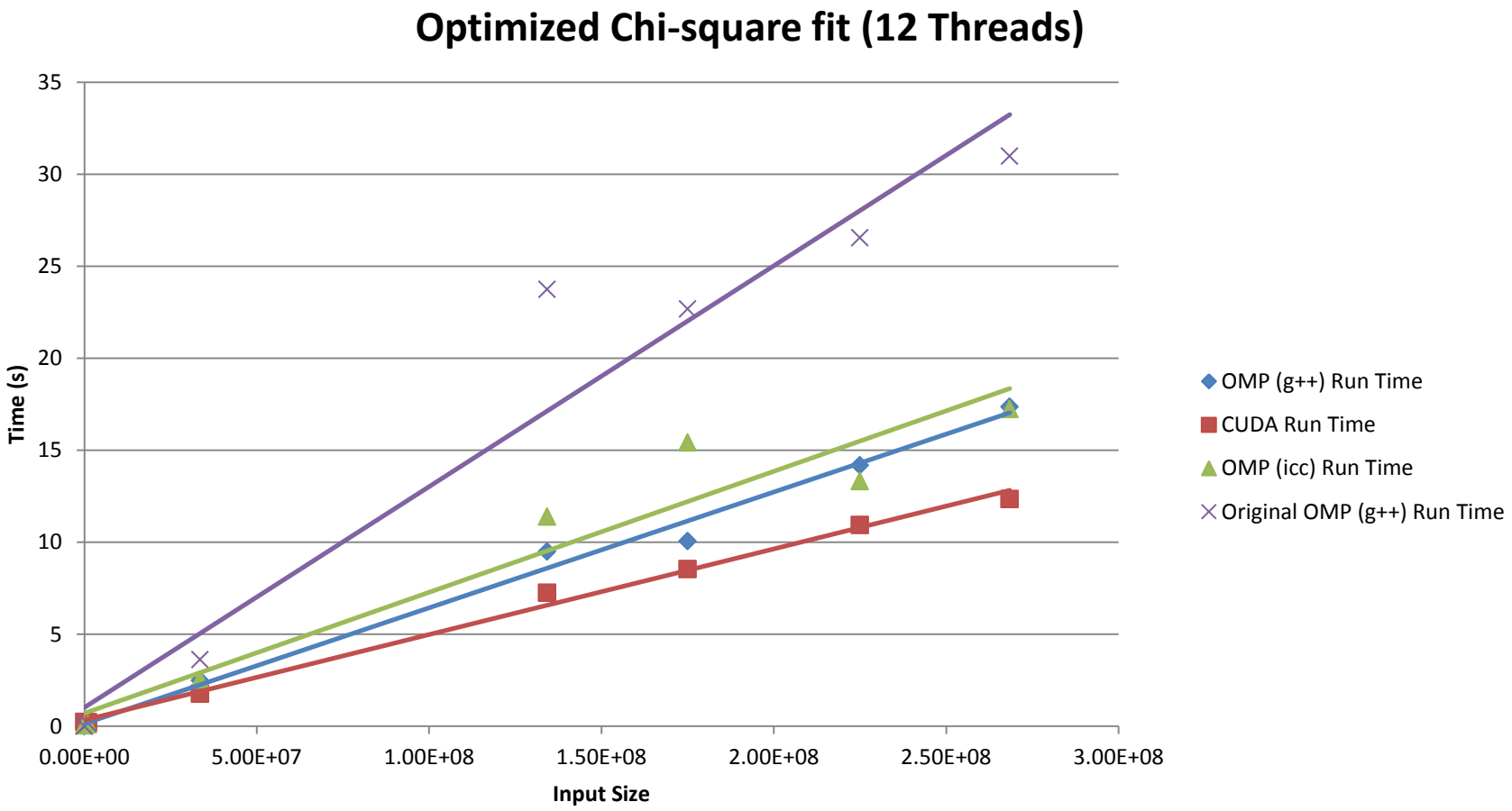
Optimizing Chi-square fit algorithm

- Instead of executing chisq /= error in each loop, chisq *= oneOverError is executed in each loop
- oneOverError is set equal to 1/error as an argument to the functor and therefore is only calculated once.

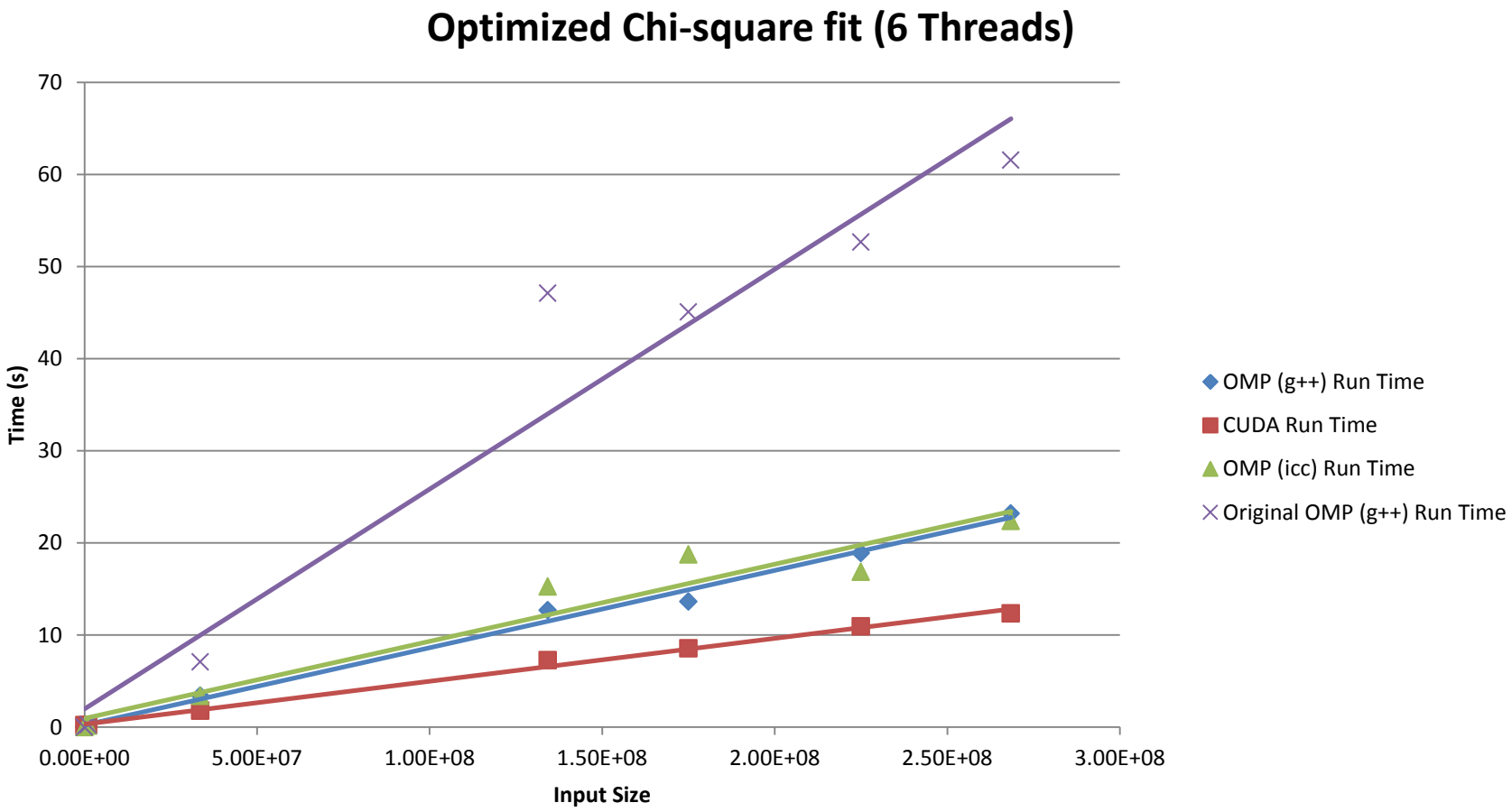
Optimized Chi-square fit: CUDA vs. OpenMP (16 Threads)



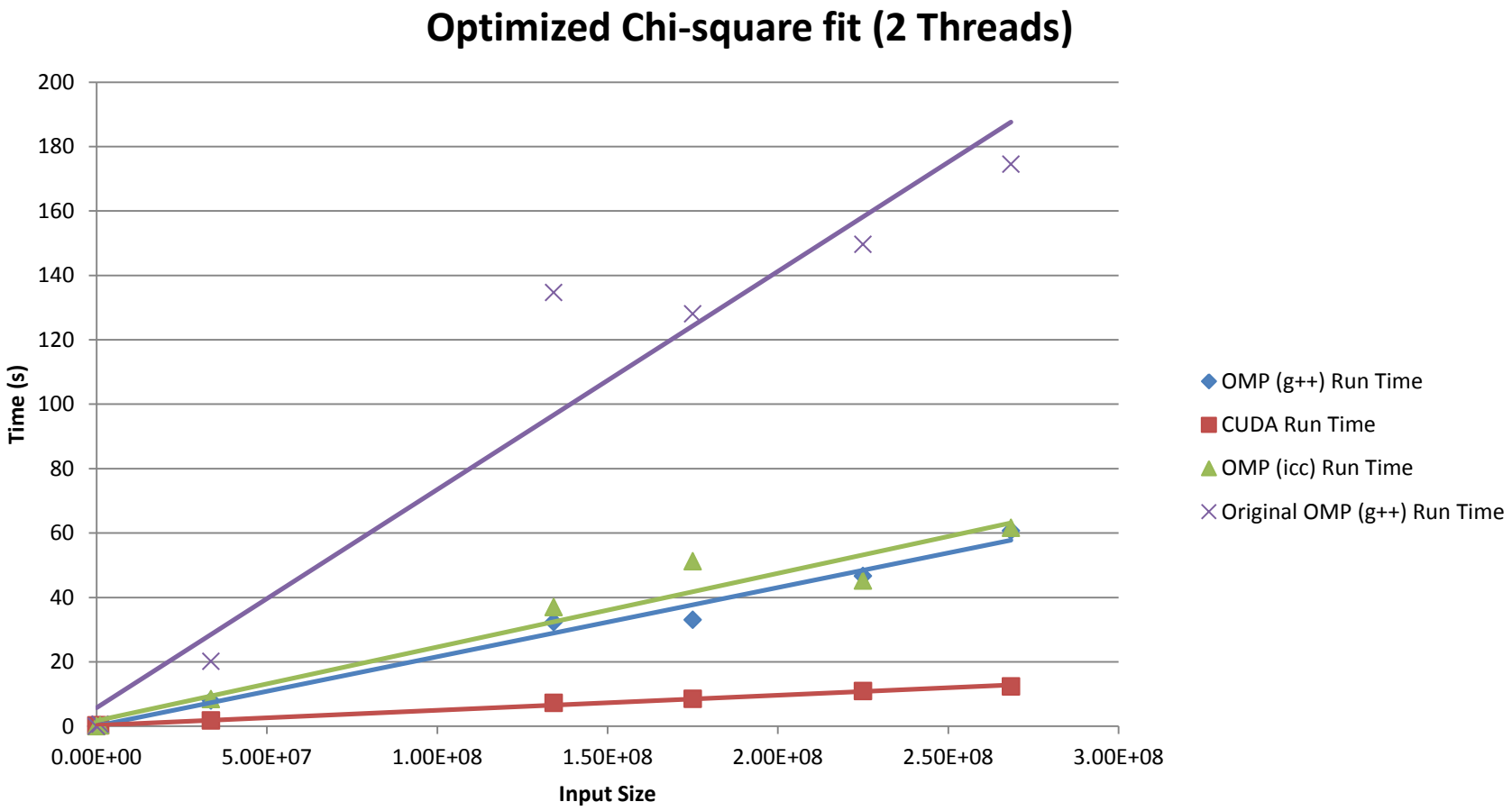
Optimized Chi-square fit: CUDA vs. OpenMP (12 Threads)



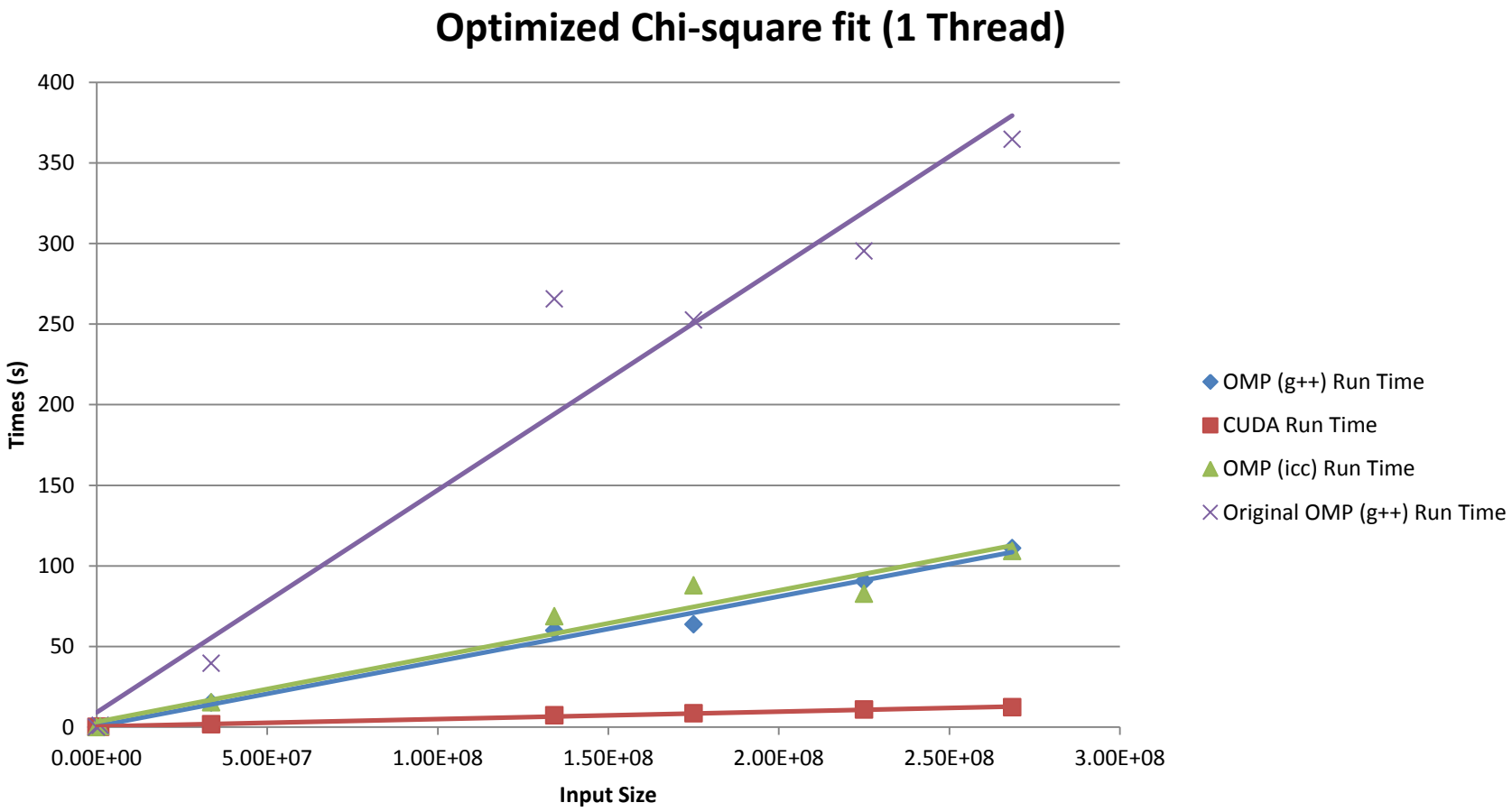
Optimized Chi-square fit: CUDA vs. OpenMP (6 Threads)



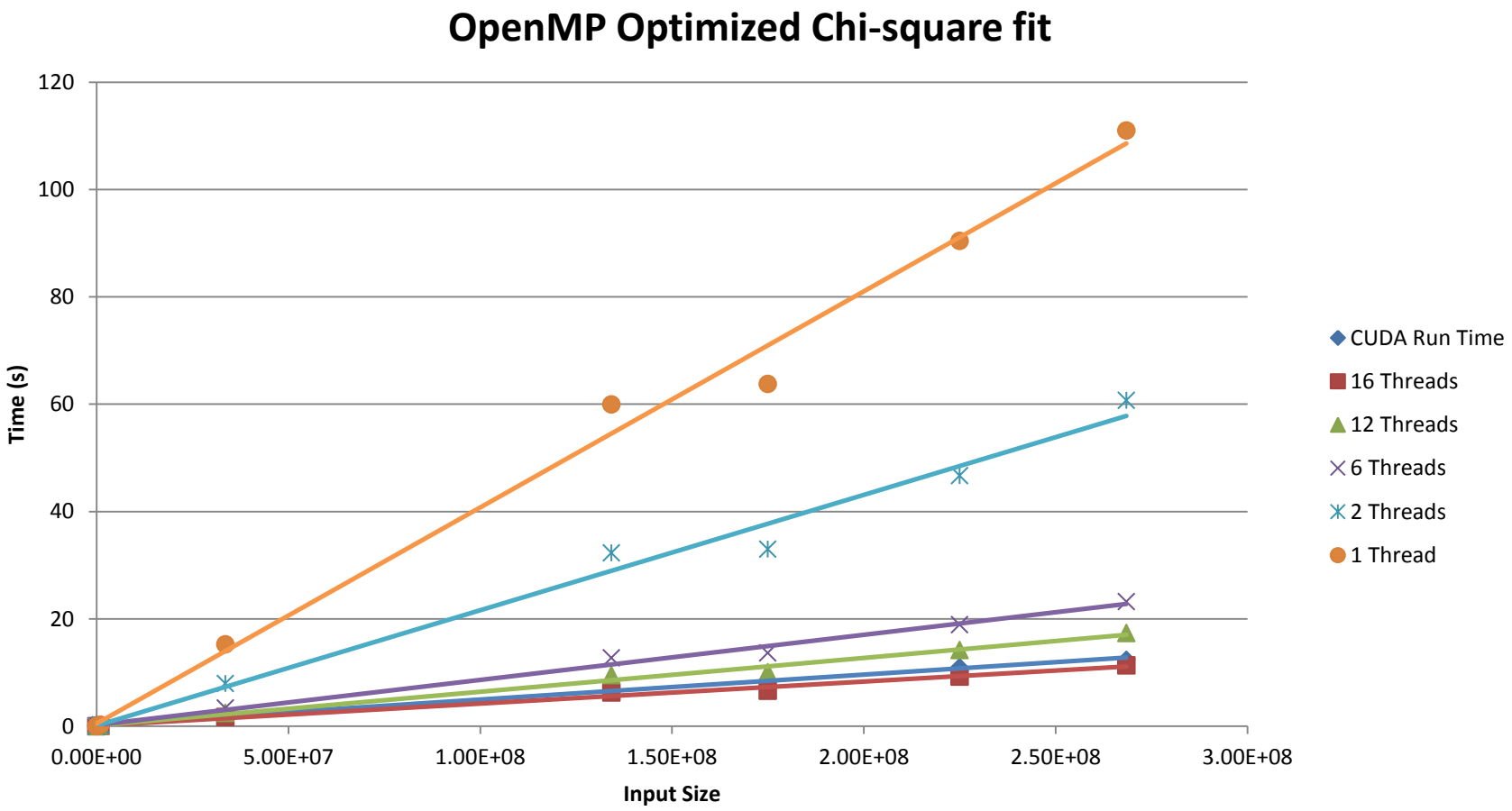
Optimized Chi-square fit: CUDA vs. OpenMP (2 Threads)



Optimized Chi-square fit: CUDA vs. OpenMP (1 Threads)



Optimized Chi-square Fit: Number of OpenMP Threads



GNU Compiler vs. Intel Compiler

GNU Compiler vs. Intel Compiler

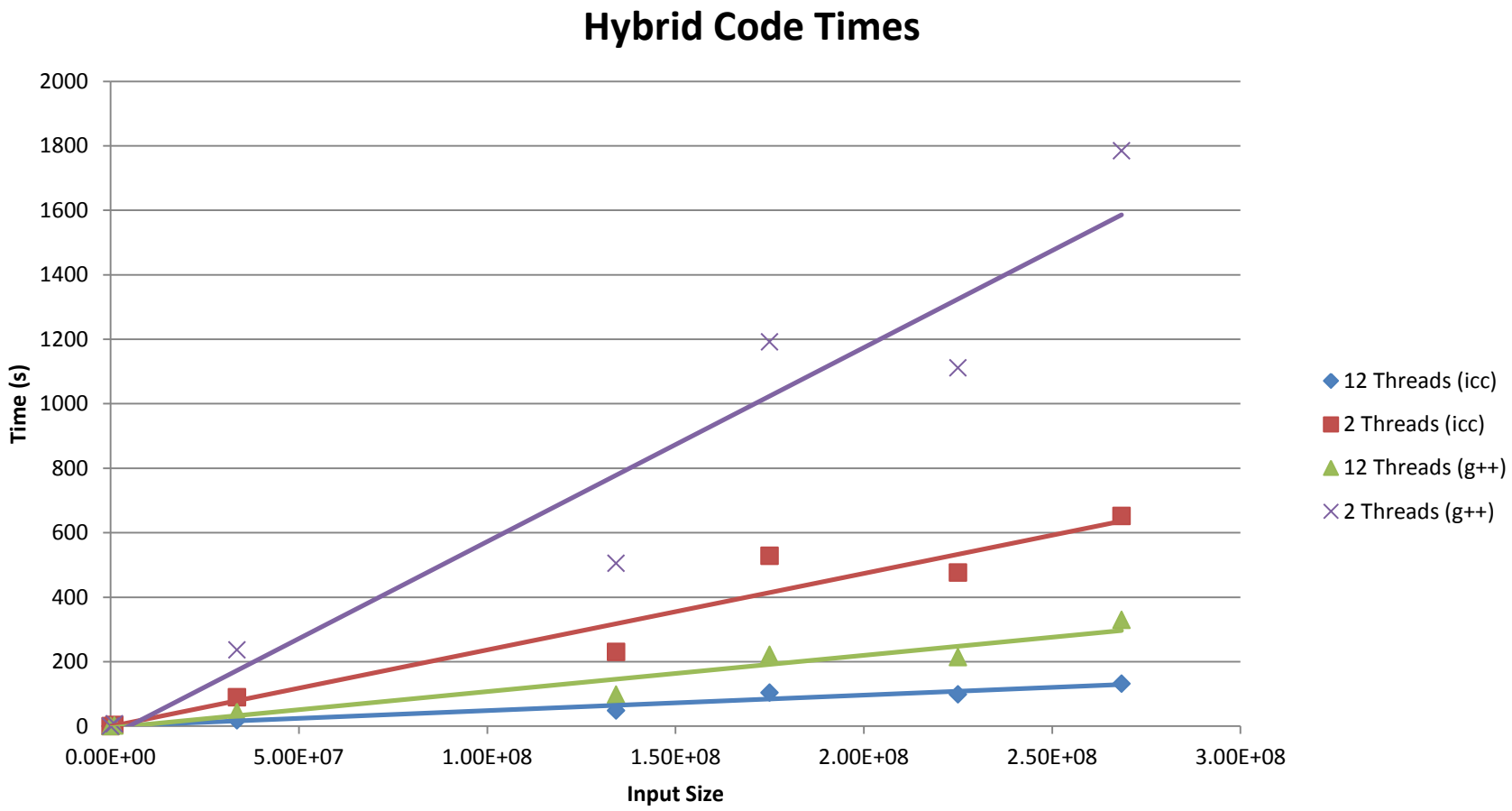
- The Gaussian fit example runs faster when compiled with the Intel compiler because the Intel compiler efficiently vectorizes the code.
- No performance difference is seen between the Chi-square fit example with the Intel compiler and the GNU compiler because the Chi-square fit example is memory bound.
- A hybrid code with the data layout of the Chi-square fit example and the mathematical work of the Gaussian fit example was used to test that the Chi-square fit example is memory bound.
- Due to the extra mathematical work, the hybrid code runs faster when compiled with the Intel compiler due to vectorization.

GNU Compiler vs. Intel Compiler (cont.)

```
__device__ double operator () (tuple<double, double> xypair)
{
   // Extract x and y from tuple
   double xval = get<0>(xypair);
   double yval = get<1>(xypair);

   // Calculate Gauss for x
   double retX = (xval - mean);
   retX /= sigma;
   retX *= retX;
   retX = exp(-0.5*retX);
   retX /= sigma;
   retX /= sqrt(2*M_PI);

   // Calculate Gauss for y
   double retY = (yval - mean);
   retY /= sigma;
   retY *= retY;
   retY = exp(-0.5*retY);
   retY /= sigma;
   retY /= sqrt(2*M_PI);
   return (0.5 * retX) + (0.5 * retY);  // Return average
 }
```

```
void chisq (int& npar, double* deriv, double& fVal,
            double param[], int flag) {
   double mean = param[0];
   double sigma = param[1];
   ChisqFunctor functor(mean, sigma);
   // Note hardcoded error!

   double initVal = 0;
   fVal = transform_reduce(
            make_zip_iterator(make_tuple(
                     dev_xvals->begin(),
                     dev_yvals->begin())),
            make_zip_iterator(make_tuple(
                     dev_xvals->end(),
                     dev_yvals->end())),
            functor,
            initVal,
            thrust::plus<double>());
}
```

GNU Compiler vs. Intel Compiler (cont.)

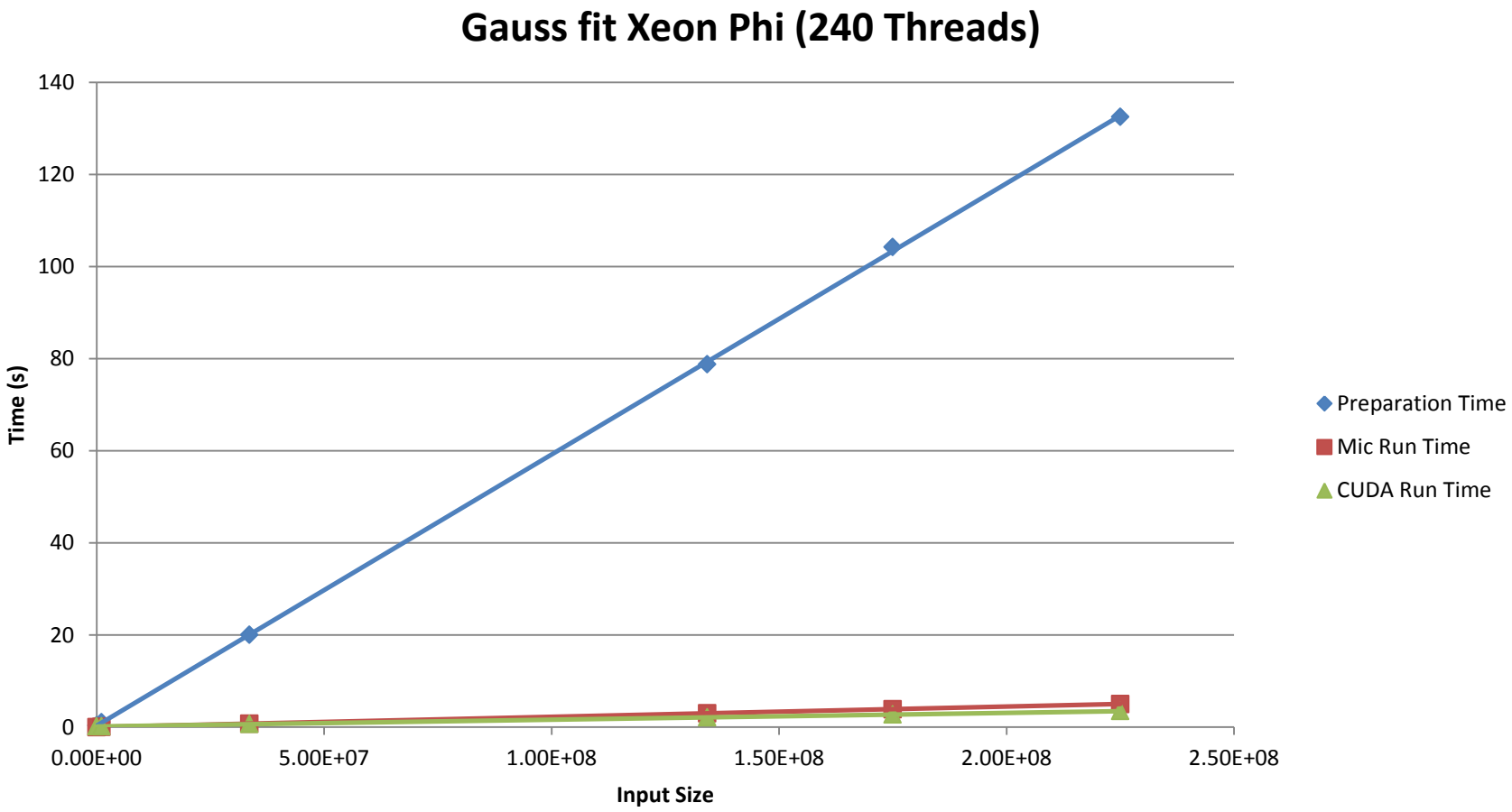


OpenMP Backend on Intel Xeon Phi (Native Execution)

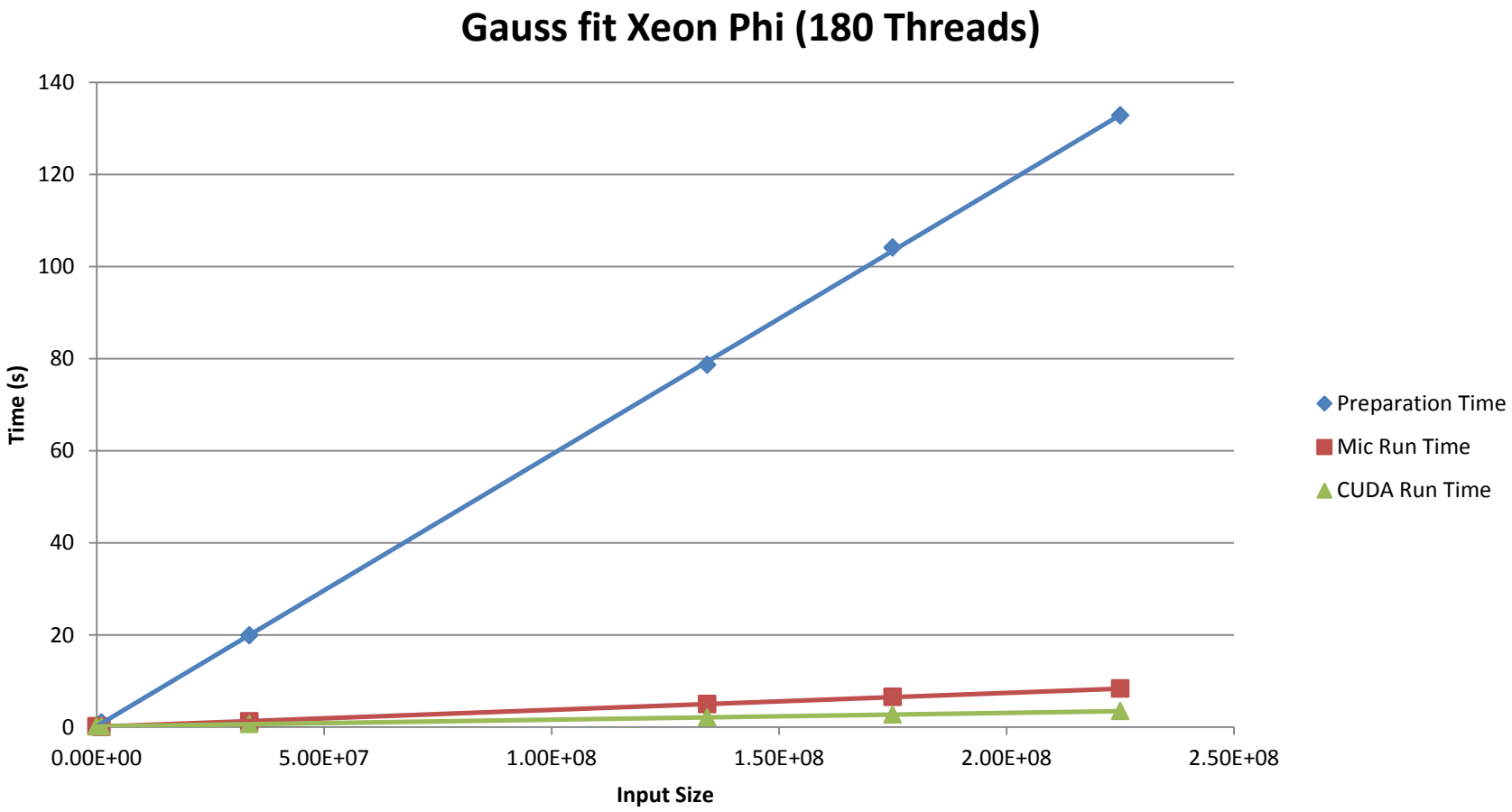
Compile (Gauss fit: Native Xeon Phi)

- Gauss Fit
  - icc solution2.cpp -O2 -o omp.out -mmic -openmp -DTHRUST_DEVICE_BACKEND=THRUST_DEVICE_BACKEND_OMP -liomp5 -I $CUDA_HOME/include -L~/local/mic/lib -L./rootstuff/ -lRootUtils -Wl,-rpath=/nfs/17/osc0724/local/mic/lib -Wl,-rpath=/nfs/17/osc0724/mic/nativeMic/solution2/rootstuff
- Rootstuff
  - icc -I. -O2 -mmic -m64 -fPIC -pthread -g -c -o TMinuit.o TMinuit.cc
  - icc -I. -O2 -mmic -m64 -fPIC -pthread -g -c -o TRandom.o TRandom.cc
  - icc -I. -O2 -mmic -m64 -fPIC -pthread -g -c -o TRandom3.o TRandom3.cc
  - icc -mmic -shared -Wl,-soname,libRootUtils.so -O2 -mmic -m64 -fPIC -pthread -g -o libRootUtils.so TMinuit.o TRandom.o TRandom3.o\

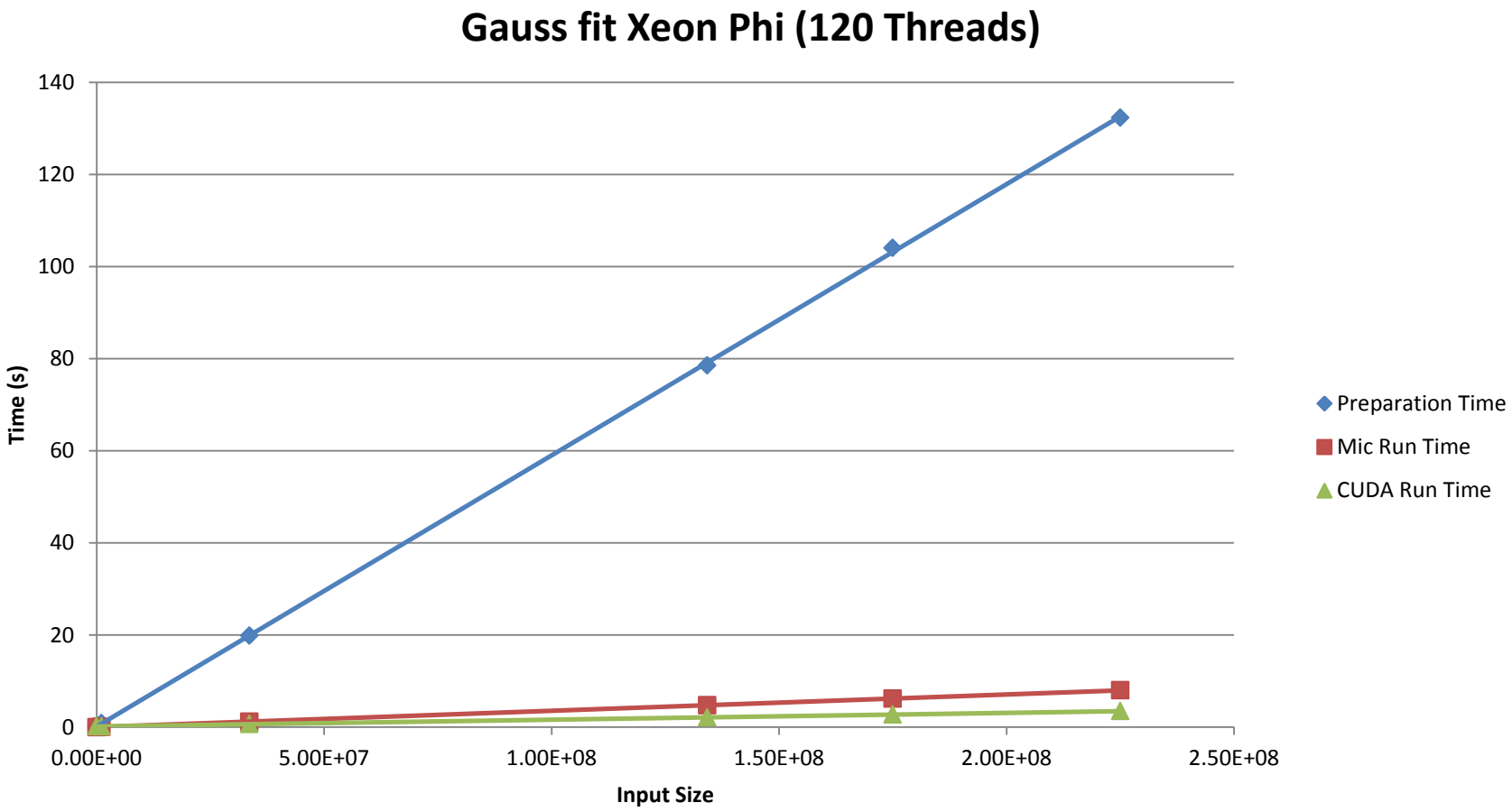
Gauss fit: Native Xeon Phi (240 Threads)



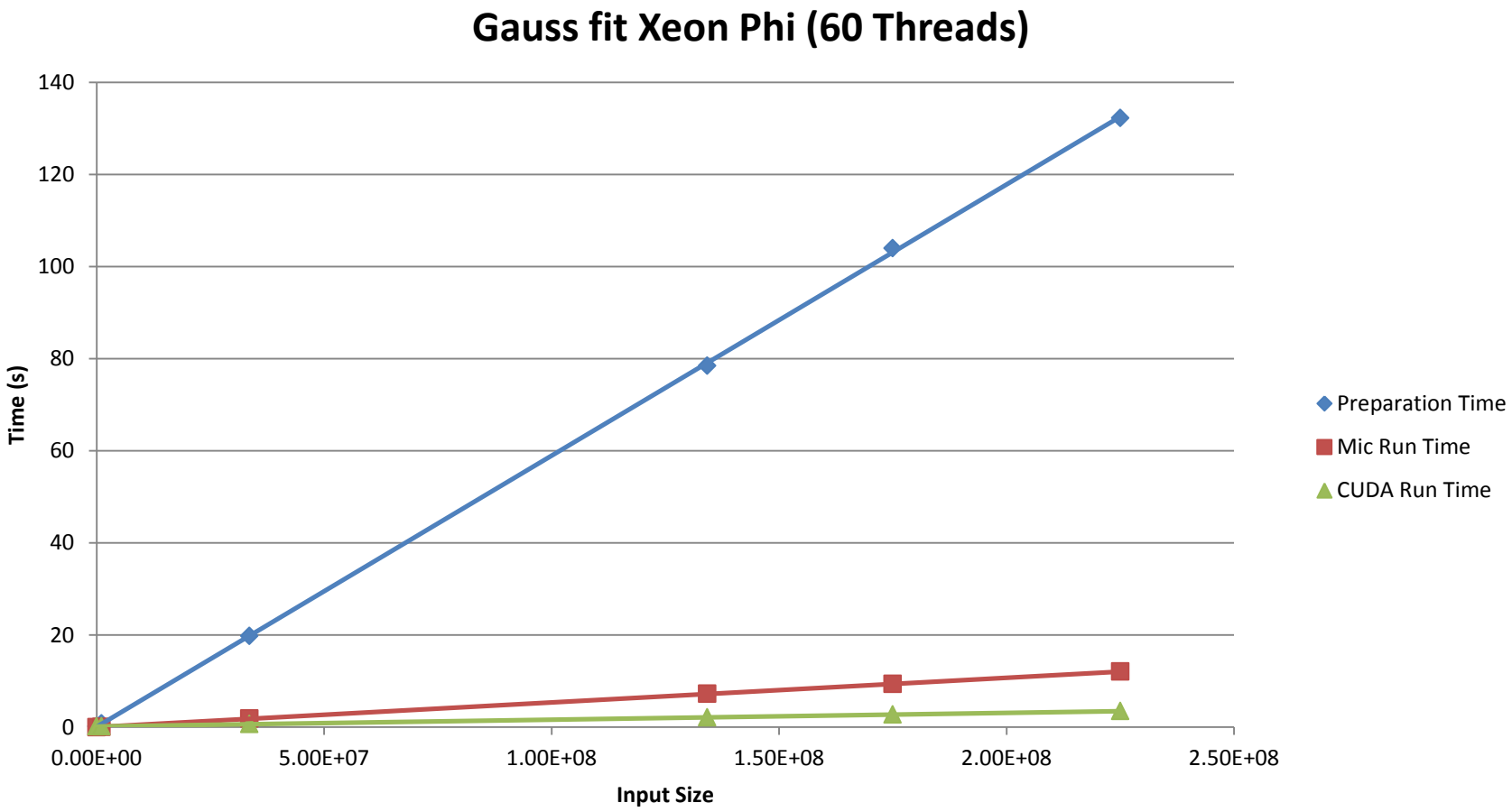
Gauss fit: Native Xeon Phi (180 Threads)



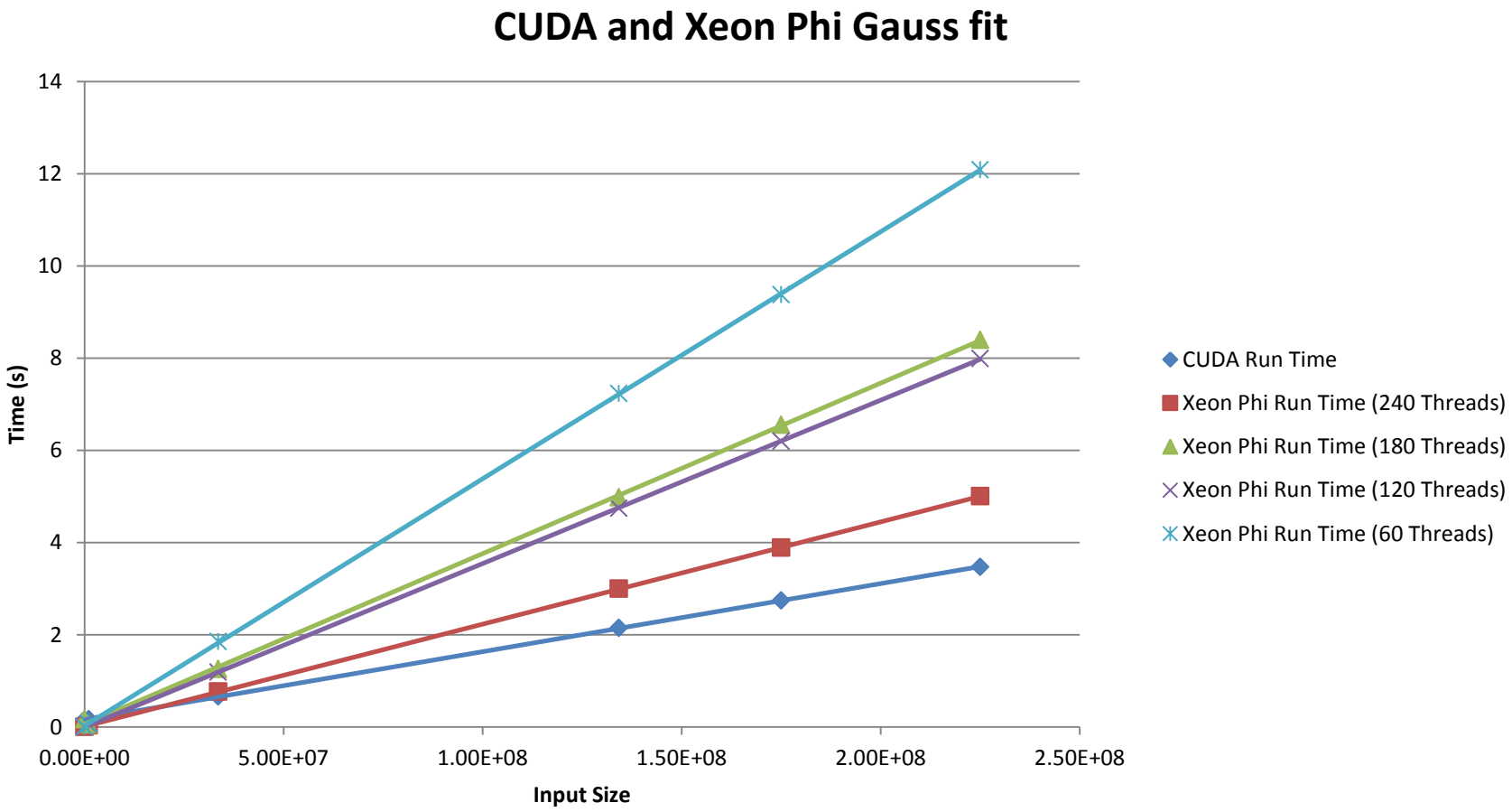
Gauss fit: Native Xeon Phi (120 Threads)



Gauss fit: Native Xeon Phi (60 Threads)



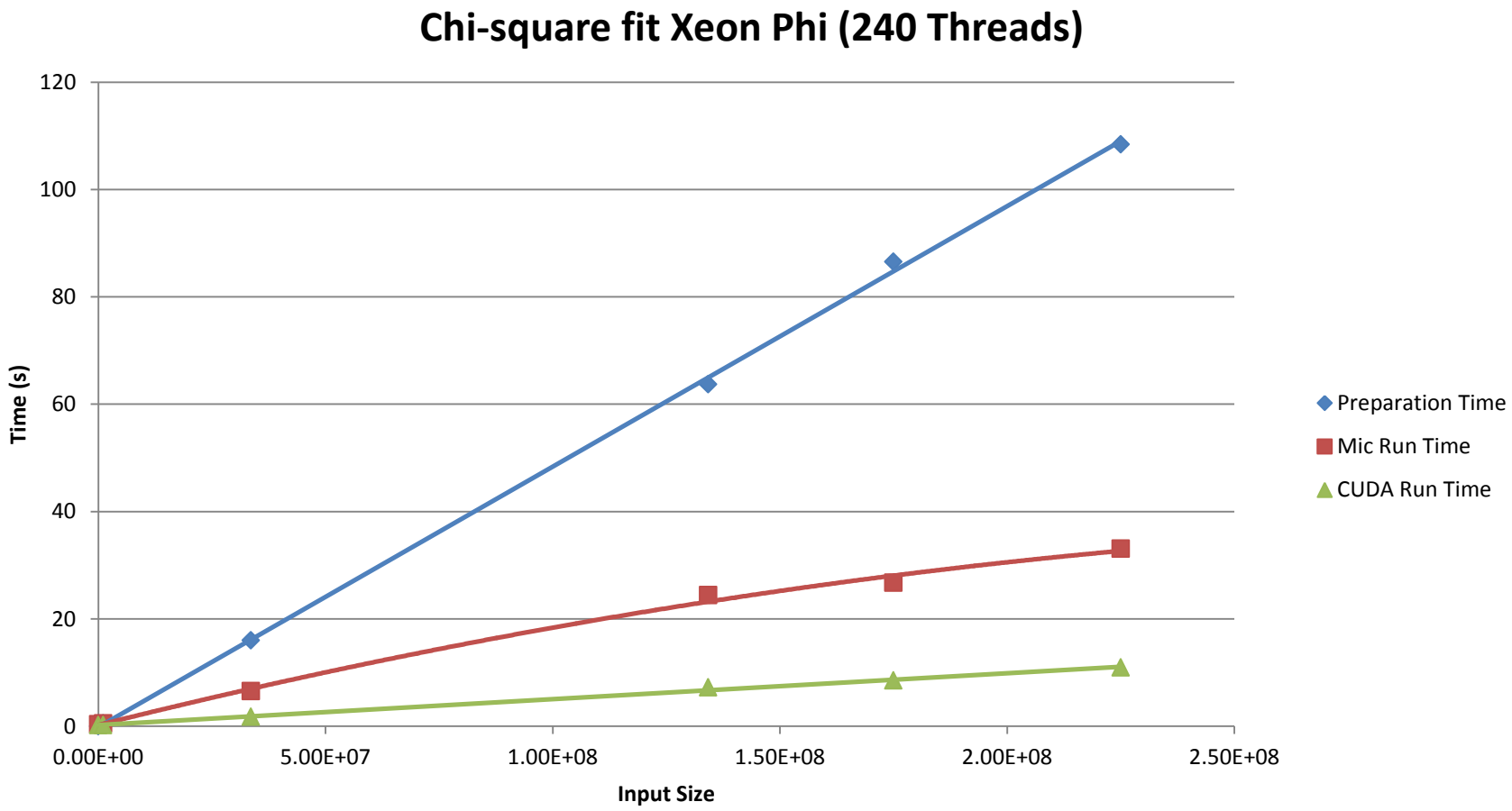
Gauss fit: CUDA vs. Native Xeon Phi



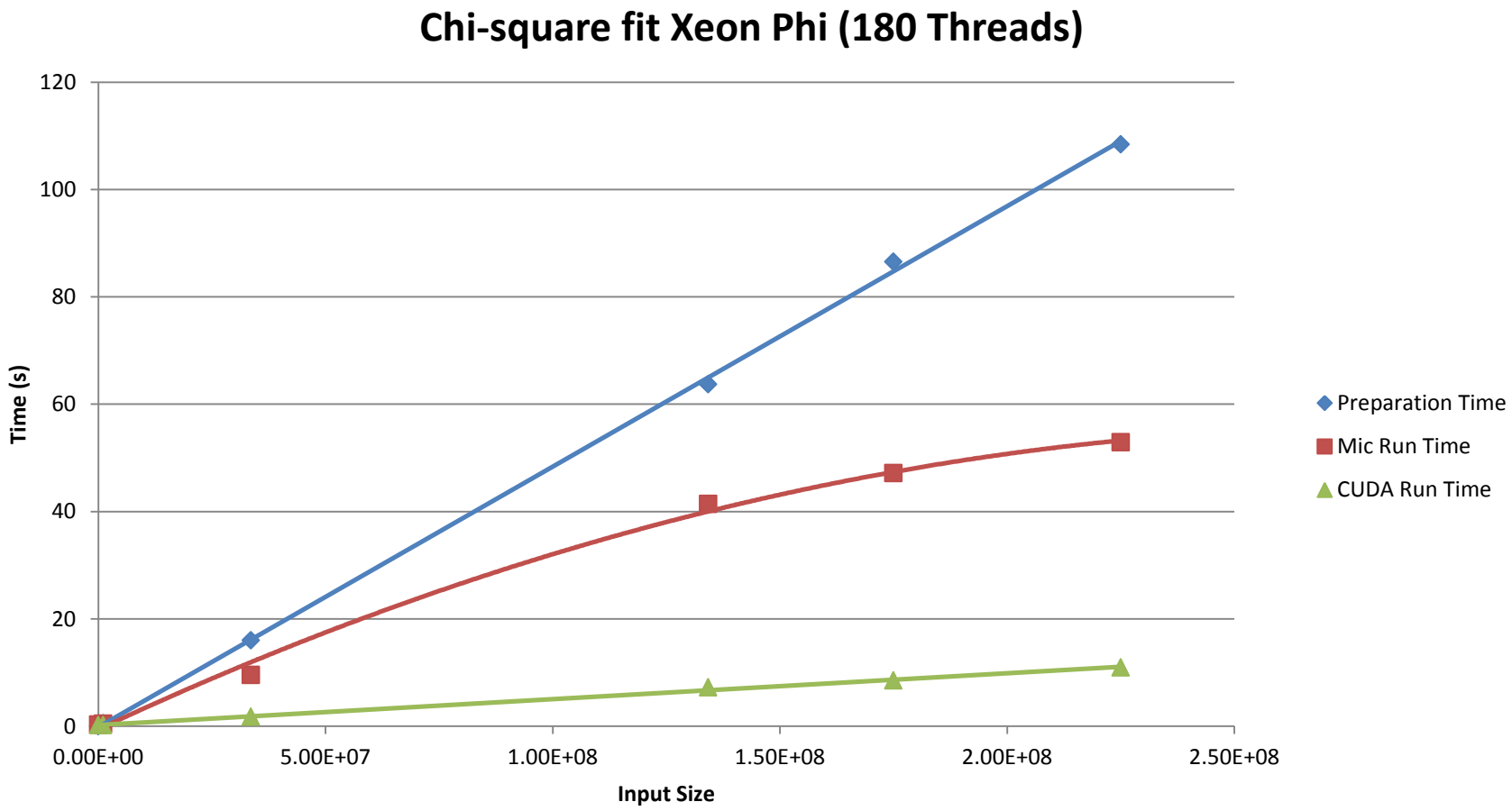
Compile (Chi-square fit: Native Xeon Phi)

- Chi-square Fit
  - icc example2.cpp -O2 -o omp.out -mmic -openmp -DTHRUST_DEVICE_BACKEND=THRUST_DEVICE_BACKEND_OMP -liomp5 -I $CUDA_HOME/include -L~/local/mic/lib -L./rootstuff/ -lRootUtils -Wl,-rpath=/nfs/17/osc0724/local/mic/lib -Wl,-rpath=/nfs/17/osc0724/mic/nativeMic/example2/rootstuff
- Rootstuff
  - icc -I. -O2 -mmic -m64 -fPIC -pthread -g -c -o TMinuit.o TMinuit.cc
  - icc -I. -O2 -mmic -m64 -fPIC -pthread -g -c -o TRandom.o TRandom.cc
  - icc -I. -O2 -mmic -m64 -fPIC -pthread -g -c -o TRandom3.o TRandom3.cc
  - icc -mmic -shared -Wl,-soname,libRootUtils.so -O2 -mmic -m64 -fPIC -pthread -g -o libRootUtils.so TMinuit.o TRandom.o TRandom3.o\

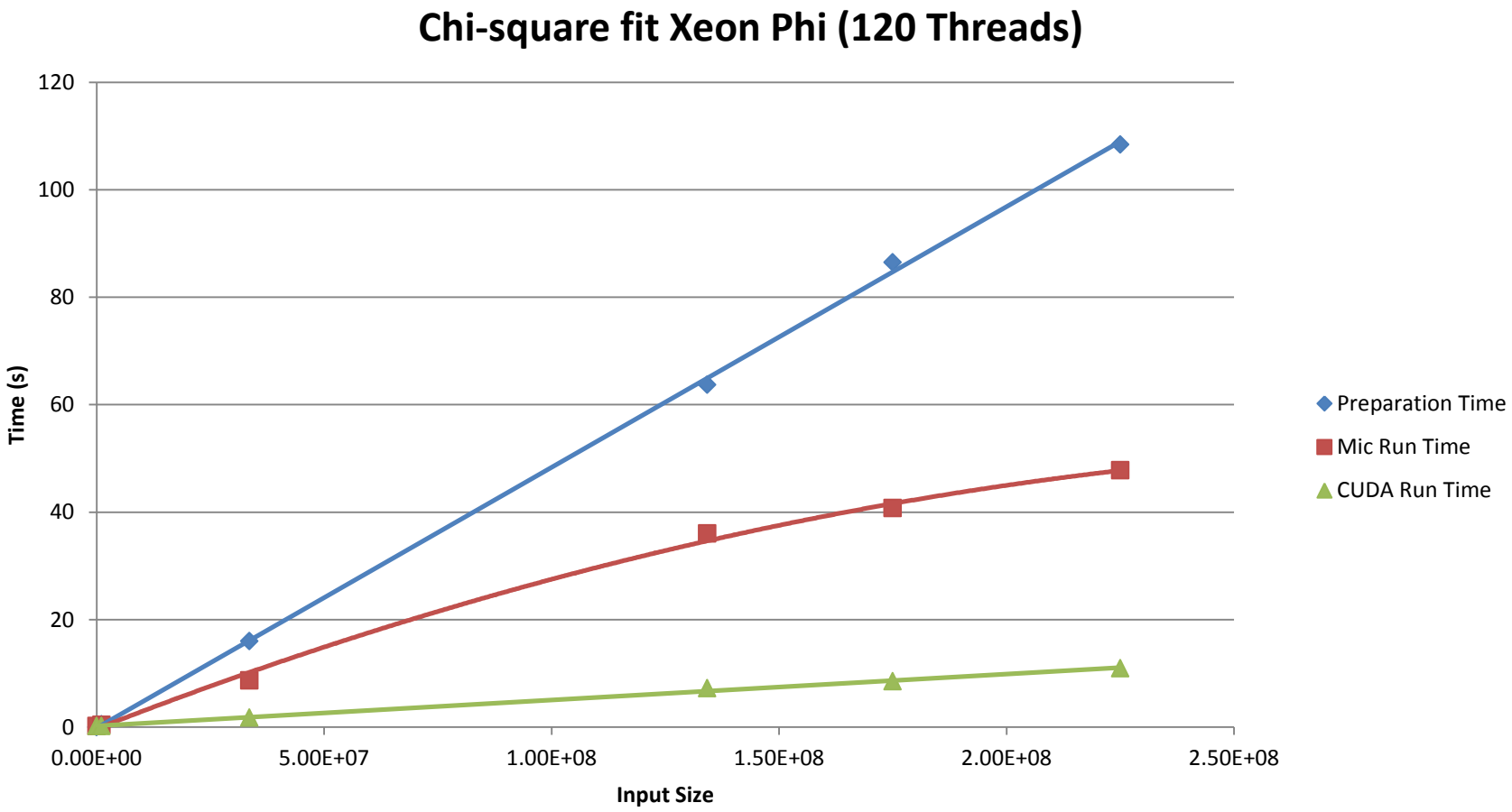
Chi-square fit: Native Xeon Phi (240 Threads)



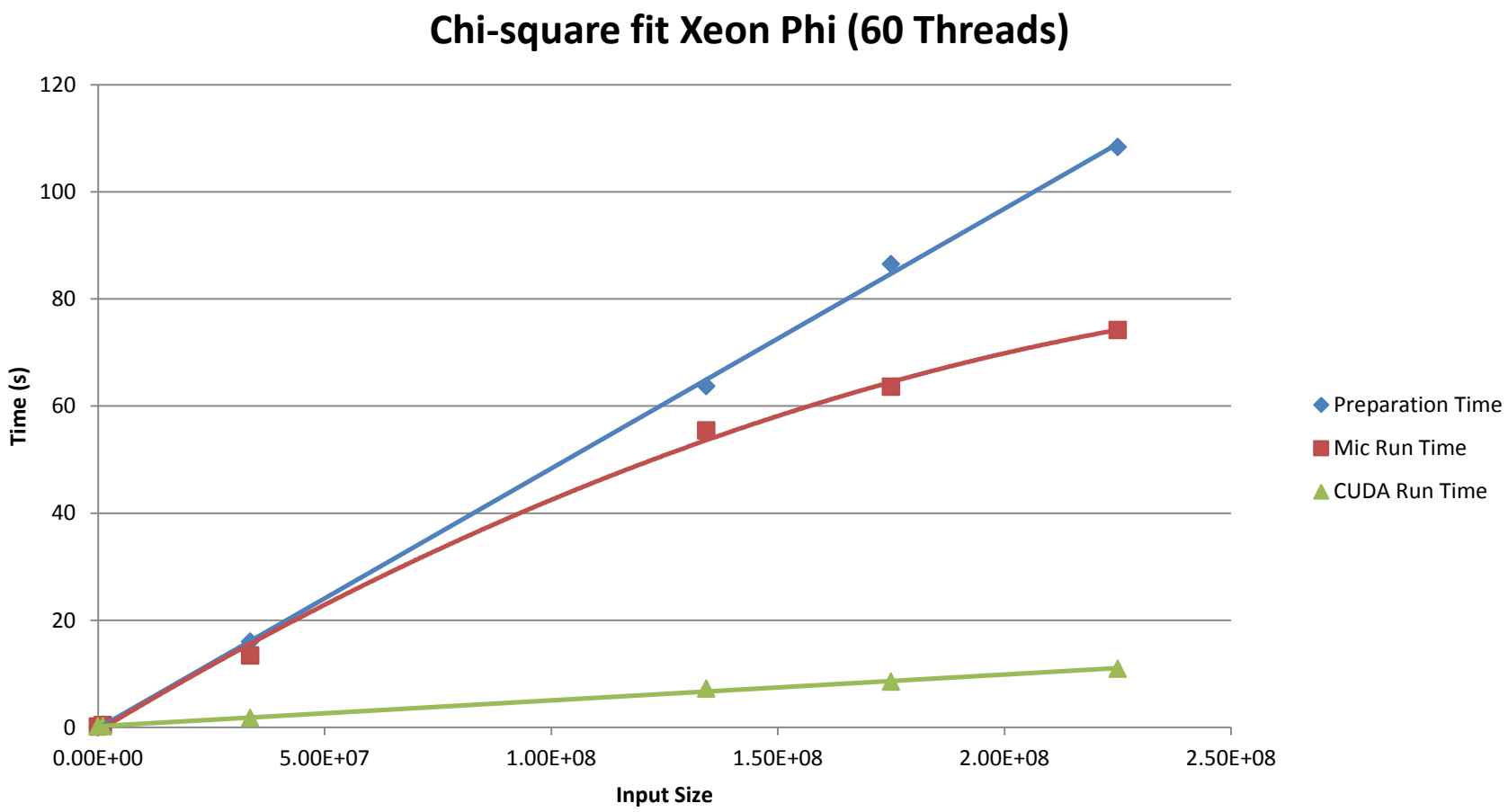
Chi-square fit: Native Xeon Phi (180 Threads)



Chi-square fit: Native Xeon Phi (120 Threads)



Chi-square fit: Native Xeon Phi (60 Threads)



Chi-square fit: CUDA vs. Native Xeon Phi