

*Exceptional  
service in the  
national  
interest*

# OpenNURBS-based Geometry Library for Next-Generation Platforms

William Roshan Quadros



## Next Generation Platforms (NGP)

- Heterogeneous architectures in memory and processors (many integrated core (MIC) and GPU devices)
- Trinity supercomputer at LANL: Cray XC30 platform with Intel Knights Landing (KNL) MIC processors
- Sierra supercomputer at LLNL: IBM platform with GPU accelerators

## NGP Challenge

- 20 year “just recompile” free ride is over! “Just recompile” could result in approximately 10x slow down
- MPI-only is no longer possible because not all cores can run MPI
- Scaling requires hybrid programming model with inter-node and on-node parallelism
- Performance portability on multiple advanced architectures is a challenge

## Kokkos

- Kokkos performance portability library preserves source code from potentially detrimental parallel directives
- Kokkos supports MPI+“X” programming model to scale on both KNL MIC-based and GPU-based next generation platforms
- Kokkos provides performant memory access patterns across multiple architectures and leverages architecture-specific features where possible
- Kokkos uses device specific backend libraries such as CUDA, pthreads, and OpenMP

## Testbed

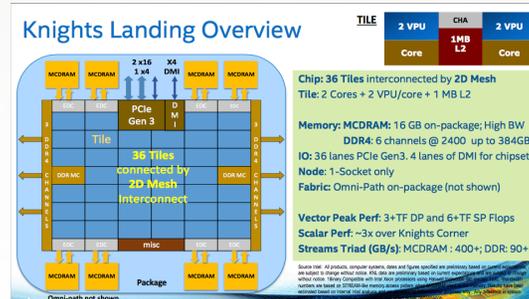


Fig 2: Trinity testbed contains KNL  
[courtesy of <http://www.hotchips.org>]

## Hybrid Programming Model

Three levels of parallelism is required:

- 1) Distributed memory parallelism via MPI
- 2) Shared memory thread level parallelism on the MIC device using Kokkos with OpenMP runtime
- 3) Vectorization for 512-bit SIMD Vector Processing Unit (VPU)

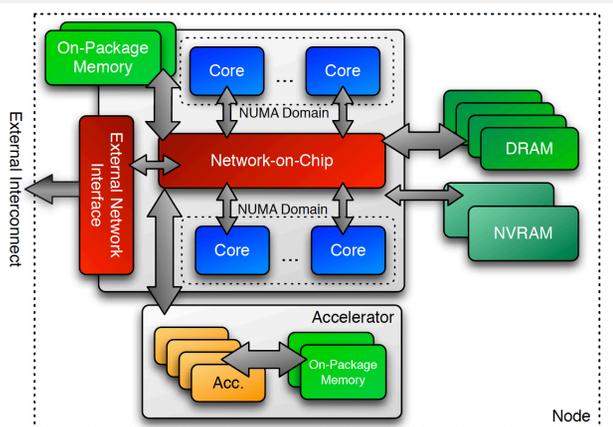


Fig 1: NGP compute node with heterogeneous cores and memory  
[courtesy of <https://github.com/kokkos>]

## Next-Gen Geometry Library

- Uses OpenNURBS as the geometry kernel
- Uses MPI + Kokkos hybrid programming model
- Supports various curves and surfaces
- Includes API for projecting points on curves and surfaces for mesh refinement

## Implementation

```
// data parallel projection of N points on a Surface
class MyClass{
private:
    int N; // N points
    ON_3dPoint *p; // array of OpenNURBS points
    ON_Surface *s; // a OpenNURBS Surface
    ...
};

MyClass::projection_method( function arguments )
{
    // 1st argument: number of points
    Kokkos::parallel_for( N, *this);
}

// operator() for Kokkos::parallel_for
MyClass::operator()( int k ) const
{
    ...
    // project Point p[k] on Surface s using OpenNURBS
    double u, v;
    s->GetClosestPoint( p[k], &u, &v );
    ON_3dPoint projected_p = s->PointAt(u, v);
}

```

## Kernels for Scaling Study

- Project points on a curve type
  - Line, Circle, Ellipse, Spline, NURBS ...
- Project points on a surface type
  - Cone, Cylinder, Sphere, Torus, NURBS ...

## KNL On-Node Performance

Kernel: Project points on a NURBS surface  
 Data size: 15,000 points  
 No. of MPI processes: 1  
 No. of threads per process: 1 to 256  
 Thread affinity: Scatter

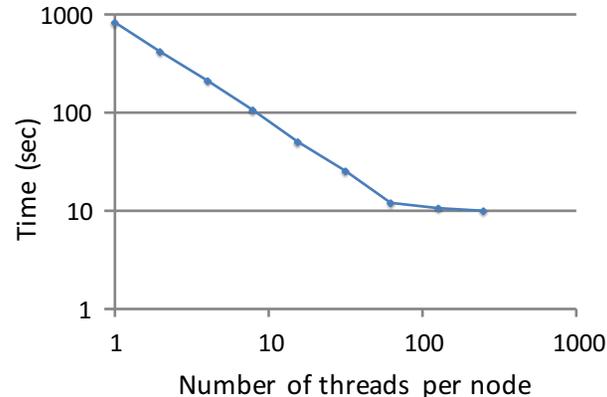


Fig 3: Speedup on KNL

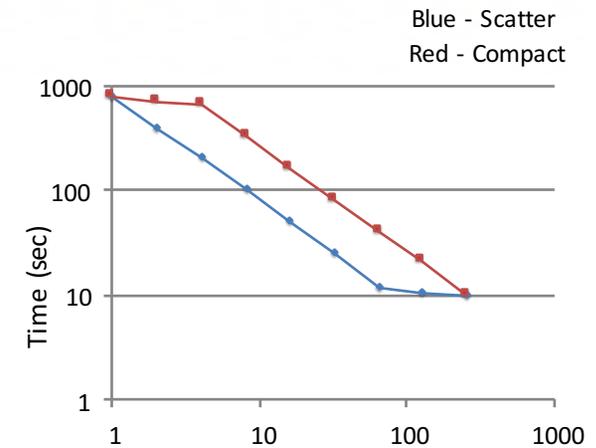


Fig 4: Thread affinity

Kernel: Project points on a surface  
 Data size: 10 to 1E6  
 No. of MPI processes: 1  
 No. of threads per process: 64  
 Thread affinity: Scatter

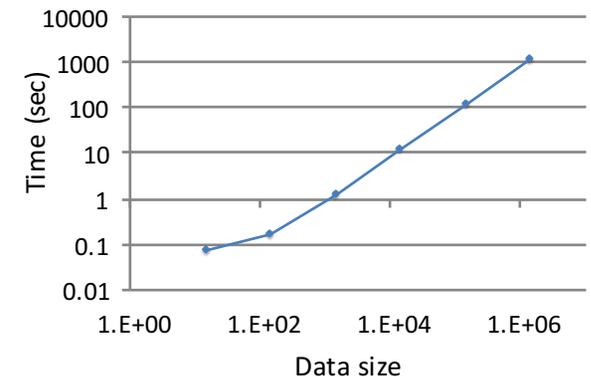


Fig 5: Near linear scaling