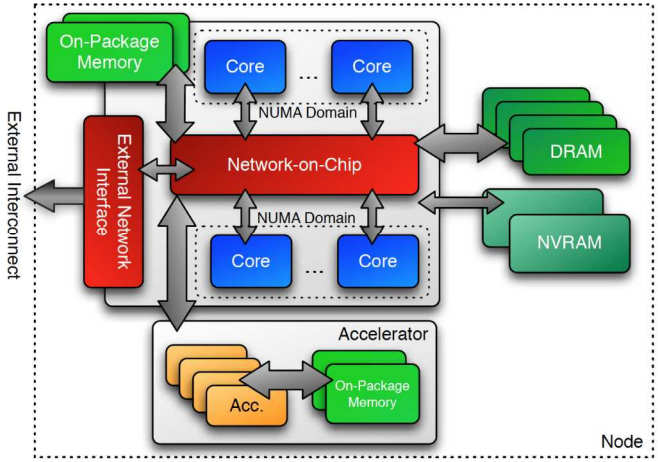
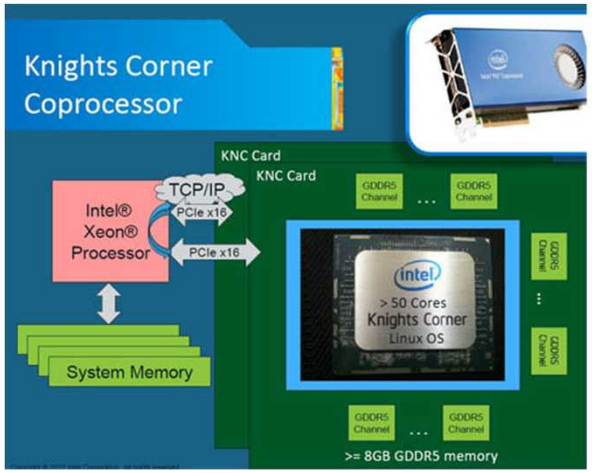
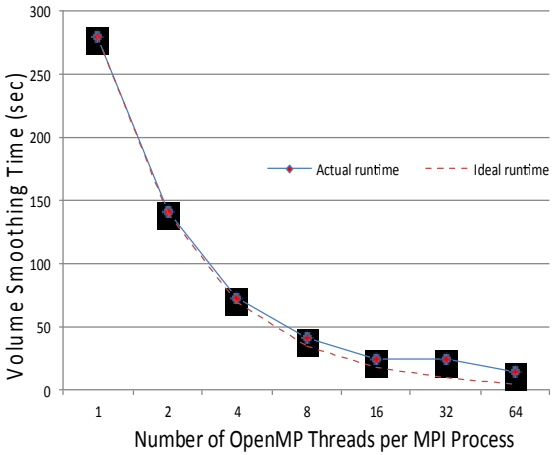
SAND2015-8785C

Exceptional service in the national interest

Sandia National Laboratories







Hybrid Programming Model to Scale Legacy Volume Smoothing on Next Generation Platforms

William Roshan Quadros

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Layout of Presentation

Sandia National Laboratories

- Next Generation Platforms (NGP)
- NGP Challenge
- Test Beds
- Procedure to Scale Legacy Code
- Hybrid Programming Model
- Performance Portability
- Case Study Kernel- Volume Smoothing
- Case Study Scaling Results

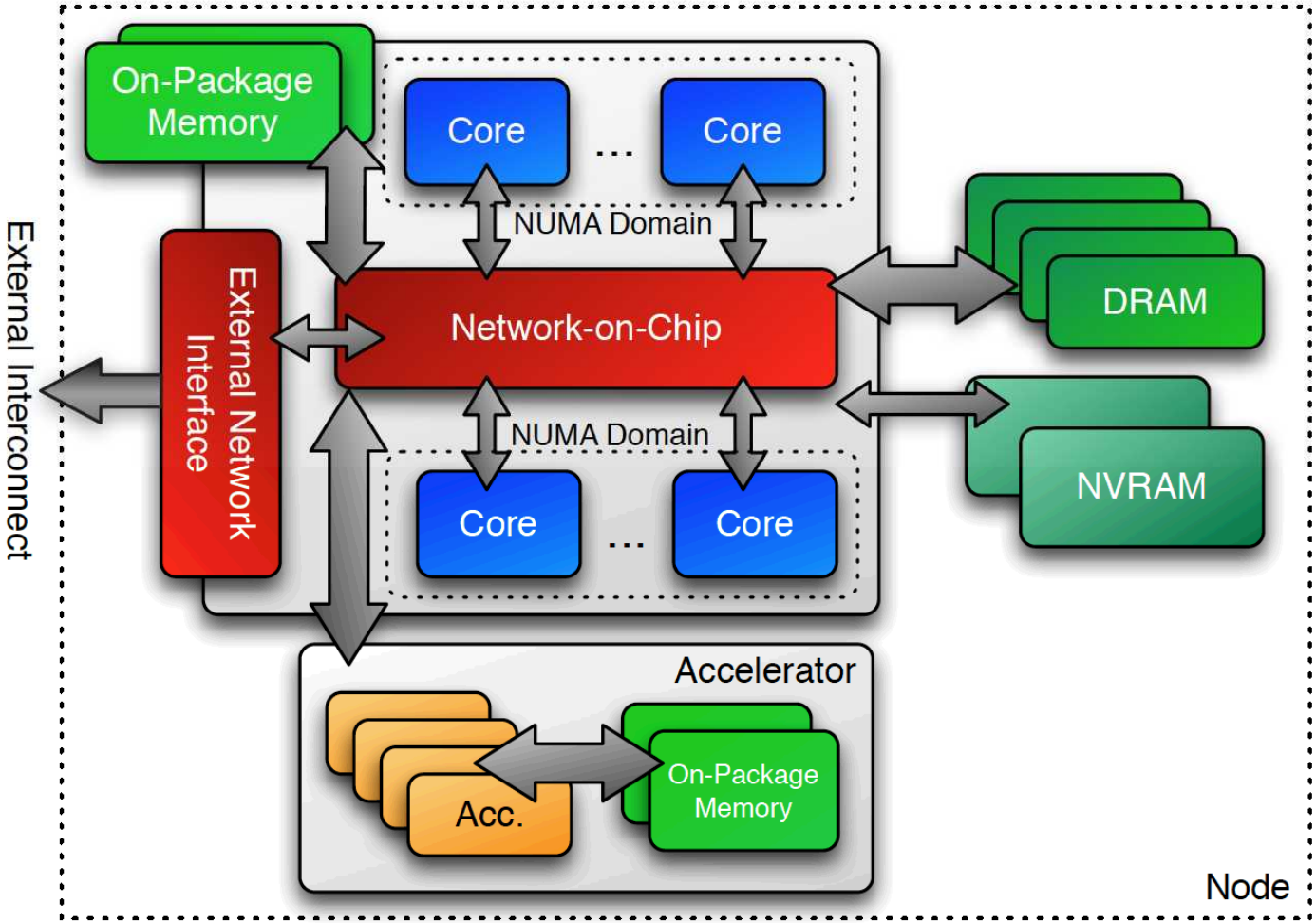
Next Generation Platforms (NGP)

Sandia National Laboratories

- The next generation platforms built towards exascale computing will have heterogeneous architectures in order to take advantage of the revolution in many integrated core (MIC) and GPU devices.
- ATS-1 Trinity Supercomputer in 2016:
  - Hosted at Los Alamos National Laboratory
  - Cray XC30 platform architecture
  - Intel Knights Landing (KNL) MIC processors
- ATS-2 Sierra Supercomputer in 2017:
  - Hosted at Lawrence Livermore National Laboratory
  - IBM platform
  - GPU accelerators.

NGP compute node with heterogeneous cores and memory

Sandia National Laboratories



Reference: https://github.com/kokkos

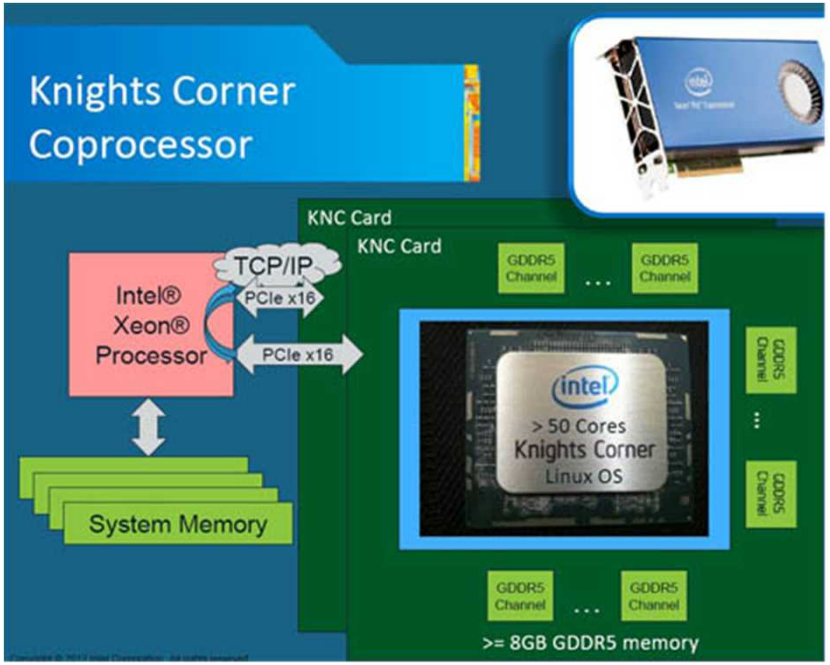
NGP Challenge

Sandia National Laboratories

- 20 year “just recompile" free ride is over! “Just recompile” would result in approx 10x slow down on NGP platforms.
- MPI-only is no longer possible because not all cores can run MPI.
- Compute nodes are heterogeneous in both cores and memory. Scaling requires leveraging node-level heterogeneous parallelism.
- Compute node architectures can be characterized by increasing thread count and decreasing memory per thread. Therefore, threading is critical for achieving high scaling.
- Performance portability on multiple advanced architectures is a challenge.
- High performance computing on these advanced architectures requires hybrid programming models containing distributed and shared memory parallelism.

Test Beds

- Test bed for Trinity supercomputer was used.
- The test bed contains Intel Knights Corners (KNC) instead of Intel Intel Knights Landing (KNL).
- Two nodes were allocated in this study. Each node contains a socket with two KNC coprocessors.
- Each KNC has 57 or 61 many integrated cores.



Intel Knight Corner (KNC)
http://ark.intel.com/products/codename/57721/Knights-Corner

Procedure to Scale Legacy Code

Sandia National Laboratories

- **STEP 1 - Profile legacy code to identify characteristics**
  - Profile to identify number of hotspots, distribution of hotspots, etc. to decide rewrite or refactor.
  - TAU has been used to profile CUBIT, and Laplace volume smoothing in CAMAL was taking 30% of the overall Sculpt meshing runtime on our test cases.

- **STEP 2: Choose a suitable programming model**
  - The next generation platforms would require both distributed and shared memory programming models.
  - Determine if data parallelism is enough or if task parallelism is required?
  - In this case study, hybrid MPI + Kokkos programming model has been selected.

Procedure to Scale Legacy Code

Sandia National Laboratories

- **STEP 3: Implement selected programming model**
  - Convert the serial code to parallel using the MPI for distributed level parallelism.
  - Make the code thread safe at the hotspots for shared memory parallelism.
  - In this study, a performance portable layer such as Kokkos as been used for thread level parallelism.

- **STEP 4: Determine optimal runtime parameters**
  - Determine the optimal number of MPI processes and threads per MPI process based on the underlying hardware architecture.
  - In this study, 4 MPI processes were specified as there are 4 KNC devices. Also, the number of threads per process ranged from 1 to 64 as each KNC contains 57 or 61 cores.

Procedure to Scale Legacy Code

Sandia National Laboratories

- **STEP 5: Optimize the code for higher scaling**

  Once the code runs on NGP, optimize the code

  - Efficient memory access
  - Reduced communication
  - Efficient data structure changes
  - Vectorization
  - Refactor algorithm
  - …

Hybrid Programming Model

Sandia National Laboratories

Three levels of parallelism is required on Trinity test beds

① Distributed memory parallelism supported through the Intel MPI library:

Distributed parallelism would require optimal domain decomposition considering load balancing and MPI communication cost.

② Shared memory thread level parallelism on the MIC device using OpenMP

Thread level parallelism on 57 or 61 core KNC would require loop-level data parallelism via a threading library. In this study, Kokkos layer was used on top of OpenMP library to achieve performance portability.

① Vectorization for the 512-bit SIMD Vector Processing Unit (VPU) of KNC.

Intel VectorAnalyzer and compiler flags/reports can be used to vectorize the code to achieve fine-grained parallelism on VPU.

Performance Portability

Sandia National Laboratories

- Preserving the source code from potentially detrimental parallel directives for multiple architectures is important for software maintenance.
- Kokkos provides a minimal overhead abstract layer that isolates user code from device specific hardware architectures. Goal is to write one implementation which compiles and runs on multiple architectures.
- Kokkos supports MPI+“X” programming model to scale on both KNC MIC-based and GPU-based next generation platforms.
- Kokkos provides performant memory access patterns across multiple architectures and leverage architecture-specific features where possible.
- Kokkos provides diverse capabilities
  - Execution spaces: CPU, GPU, MIC, PIM, …
  - Memory spaces: HBM, DDR w/NUMA, NVRAM, scratch pads / managed cache layers, …
  - Backend Libs: OpenMP/ACC/CL, Cuda, TBB, C++AMP, Thrust, HPX, StarPU, …

Example: Kokkos Loop-level Parallelism

Sandia National Laboratories

```
// data parallelism on N nodes
MyClass::class_method( function arguments )
 {  ...
   // 1st argument: number of nodes
   // 2nd argument: this object
   Kokkos::parallel_for( N, *this);
   ..
 }
 // operator() for Kokkos::parallel_for
MyClass::operator()( int k ) const
 {  ...
   // Laplace smoothing at node k given by Eq (1)
   laplacian_smooth_at_node(k);
  }
```

For more details visit https://github.com/kokkos/kokkos-tutorials

Case Study Kernel: Volume Smoothing

Sandia National Laboratories

- In this case study, CAMAL's Laplace volume smoothing algorithm was used as the hotspot kernel
- Laplace smoothing is given by:

$$x_{i+1,k} = \frac{1}{N}\sum_{j=1}^{N} x_{i,j}$$

N - number of adjacent nodes to node k

$x_{i,j}$ - coordinate of jth adjacent node of node k in the i^{th} iteration

$x_{i+1,k}$ – coordinate of node k in $i+1^{th}$ iteration

Case Study Scaling Results

Sandia National Laboratories

Test Case

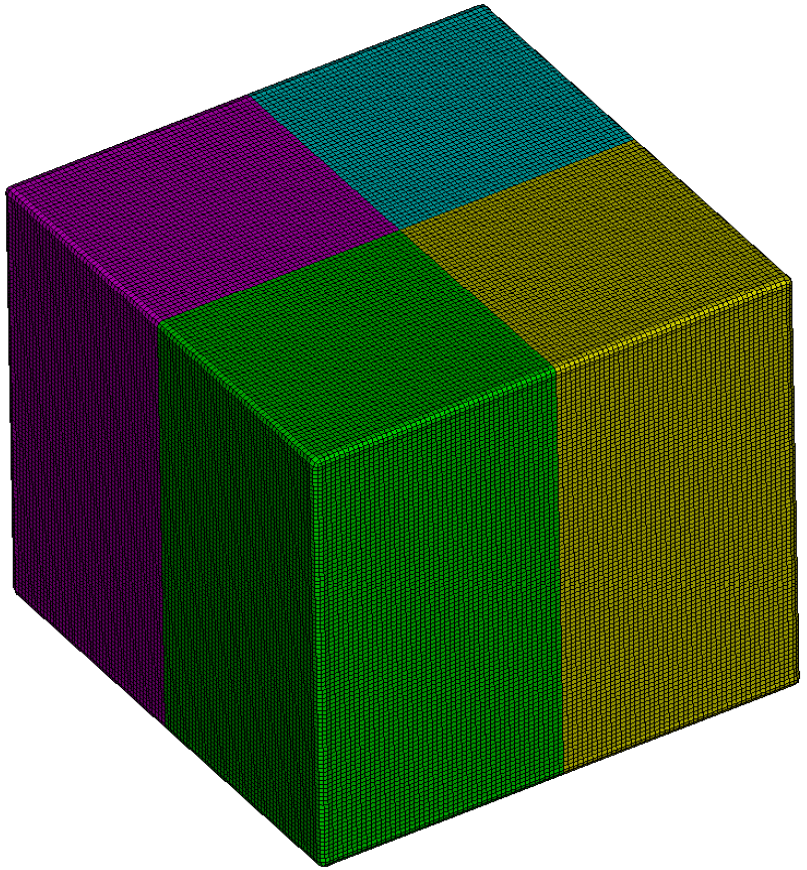
Kernel: Laplace volume smoothing

Iterations: 10

Data size: 5 million nodes

No. of MPI processes = 4

No. of threads per process = 1 to 64



Case Study Scaling Results

Sandia National Laboratories

One of the studies focused on node-level threading performance on a KNC MIC device. The MPI-only version shown in row 1 is regarded as the baseline application.

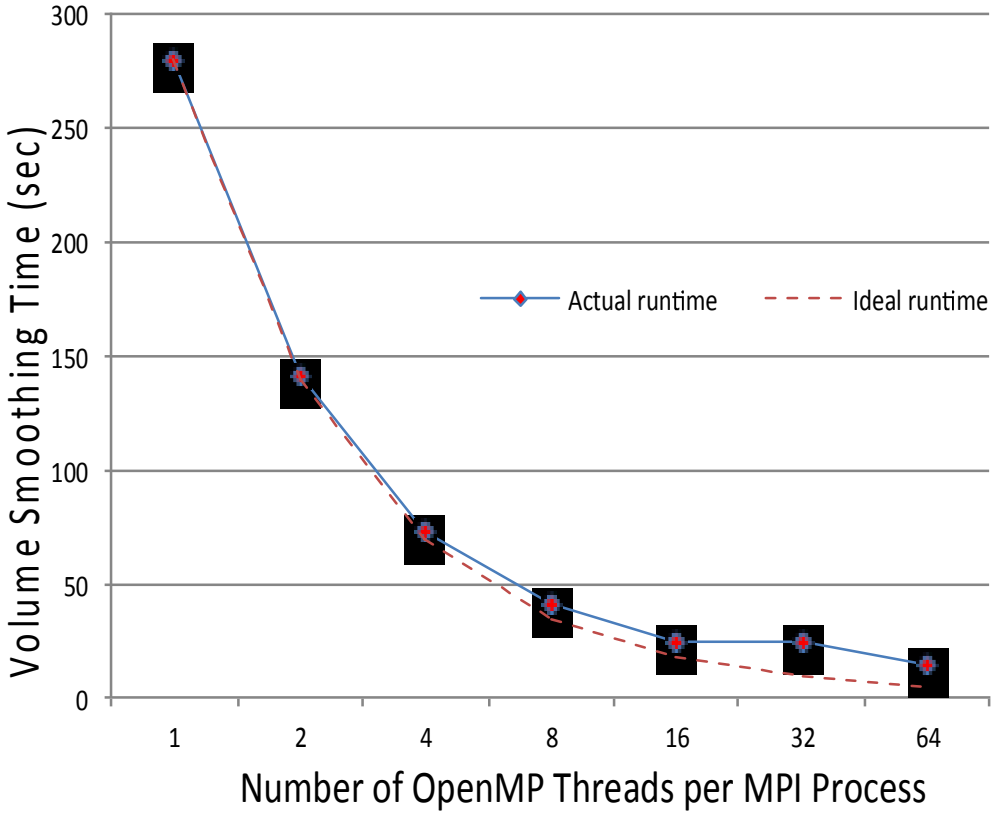
Table 1. 5-trial average result of volume smoothing on a 5 million node hex mesh

Number of Processes	Thread per Process	Process X Thread	Actual Runtime (sec)	Ideal Runtime (sec)	Percentage Deviation
4	1	4	278.88	278.88	0%
4	2	8	140.48	139.44	0.74%
4	4	16	73.22	69.72	5.02%
4	8	32	40.23	34.86	15.40%
4	16	64	24.16	17.43	38.61%
4	32	128	23.64	8.71	171.25%
4	64	256	14.24	4.35	226.79%

Case Study Scaling Results

Sandia National Laboratories

- As we increase the threads per process, the deviation from the linear scaling increases due to thread startup and overhead costs
- On a MIC device, 95% reduction in runtime (a 20X speedup) is observed, as the single thread runtime of 278.88 seconds is dropped to 14.24 seconds



Linear scaling and actual scaling

Sandia National Laboratories

Thank You!