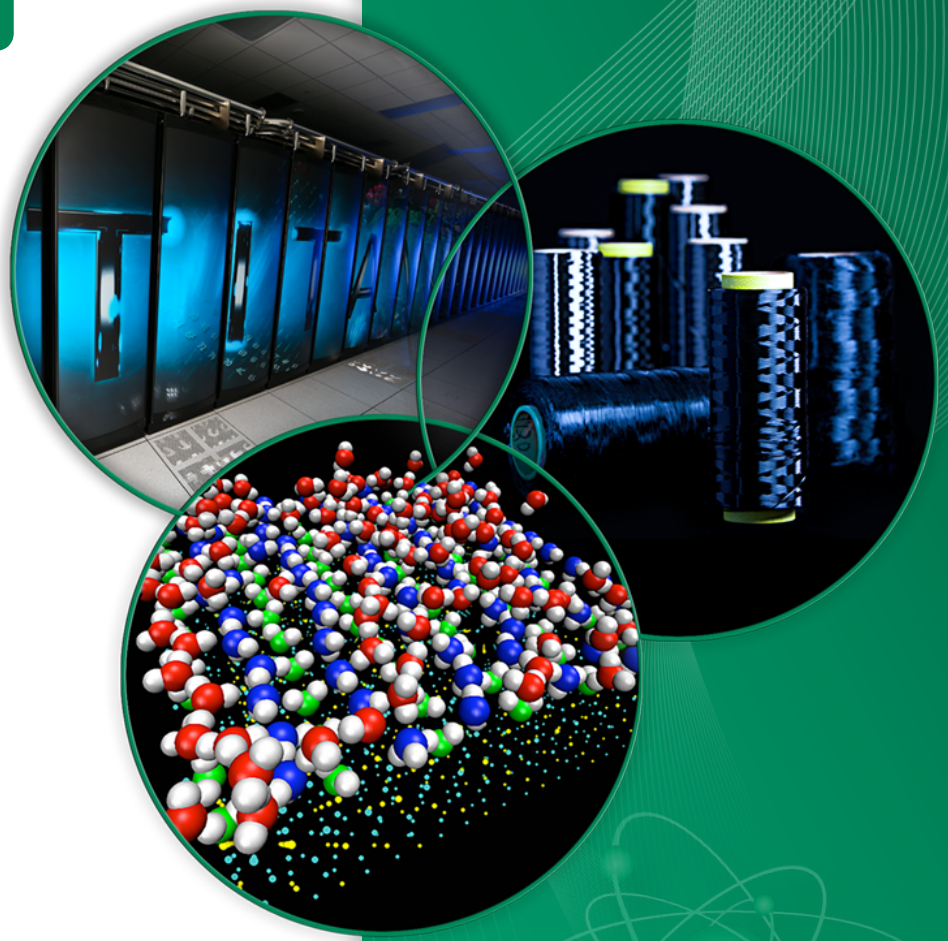
Hybrid Programming Using OpenSHMEM and OpenACC

SSCA3 Case Study



Mathew Baker, ORNL

Oscar Hernandez, ORNL

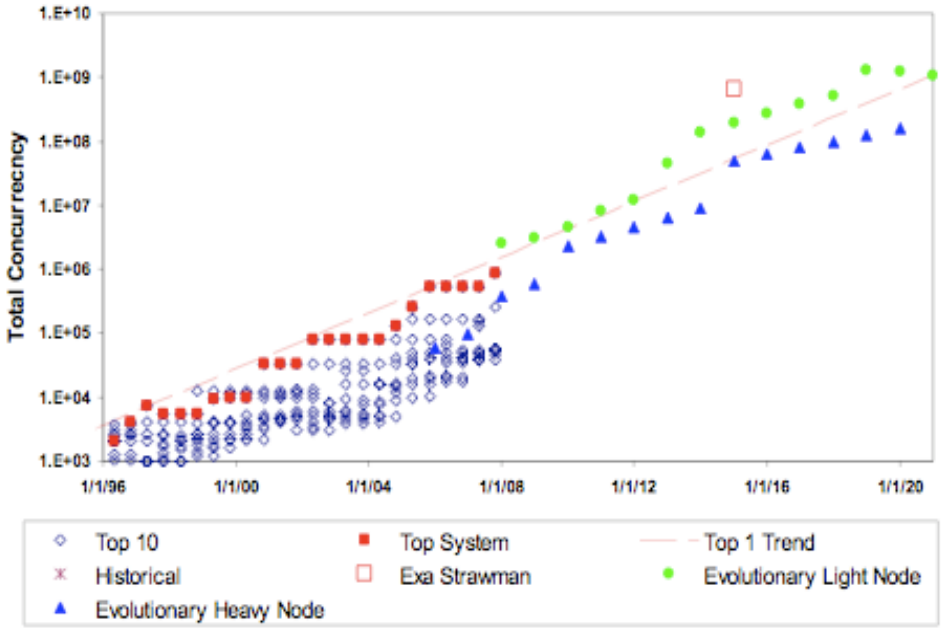
Jean-Charles Vasnier, CAPS

Maximilien de Couasnon, CAPS

CAPS

Accelerating the Path to Exascale

- Exascale expectations
  - **Extreme levels of parallelism (1B+)**
  - **Heterogeneity, either on or off die**
  - Lower memory footprint per core
  - **Deeper memory hierarchies**
  - **High cost of data movement**
  - Component failure is the norm
  - Extreme power constraints
- Accelerators are of increasing interest because they offer high ratios of performance to power
- Top systems based on accelerators
  - 8) Stampede, Xeon Phi, 2.6 PF
  - **6) Piz Daint, NVIDIA Kepler, 2.7 PF**
  - **2) Titan, NVIDIA Kepler, 17.6 PF**
  - *1) Tianhe-2, Xeon Phi, 33.8 PF*

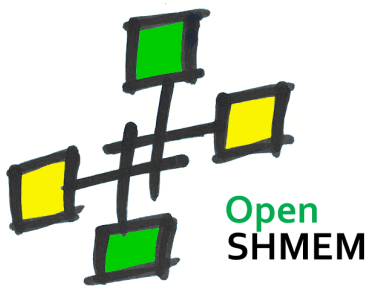


	2010	2018	Factor Change
System peak	2 Pf/s	1 Ef/s	500
Power	6 MW	20 MW	3
System Memory	0.3 PB	10 PB	33
Node Performance	0.125 Gf/s	10 Tf/s	80
Node Memory BW	25 GB/s	400 GB/s	16
Node Concurrency	12 CPUs	1,000 CPUs	83
Interconnect BW	1.5 GB/s	50 GB/s	33
System Size (nodes)	20 K nodes	1 M nodes	50
Total Concurrency	225 K	1 B	4,444
Storage	15 PB	300 PB	20
Input/Output bandwidth	0.2 TB/s	20 TB/s	100

Future Exascale Systems

- Research / co-design of future programming models and runtimes with hardware.
  - Distributed / Distributed-shared memory models
    - MPI / OpenSHMEM, UPC, Fortran 2008 CoArrays, Chapel
  - Shared Memory Models
    - OpenMP, Pthreads
  - Heterogeneous models
    - OpenACC, OpenCL, OpenMP 4.0
  - Hybrid Heterogeneous Models
    - OpenSHMEM/OpenACC, OpenSHMEM/OpenMP, MPI/OpenACC, etc
- How to efficiently map the model to the hardware while meeting application requirements?
- Need to evolve models to the future.

OpenSHMEM



- SHared MEMory library specification that implements distributed shared memory model.
  - www.openshmem.org, 1.1 spec
- OpenSHMEM is a 1-sided communications library
  - C and Fortran PGAS programming model
  - Uses symmetric data objects to efficiently communicate across processes
  - Point-to-point and collective routines
  - Synchronizations, Atomic operations
- Advantages:
  - Irregular applications, latency-driven communication
  - Small/medium size messages communication
  - Maps really well to hardware/interconnects

OpenSHMEM: Key Concepts

- Processing Element (PE) is an OpenSHMEM process
- PEs have symmetric memory
  - Static/Global variables
  - Symmetric heap

OpenSHMEM code

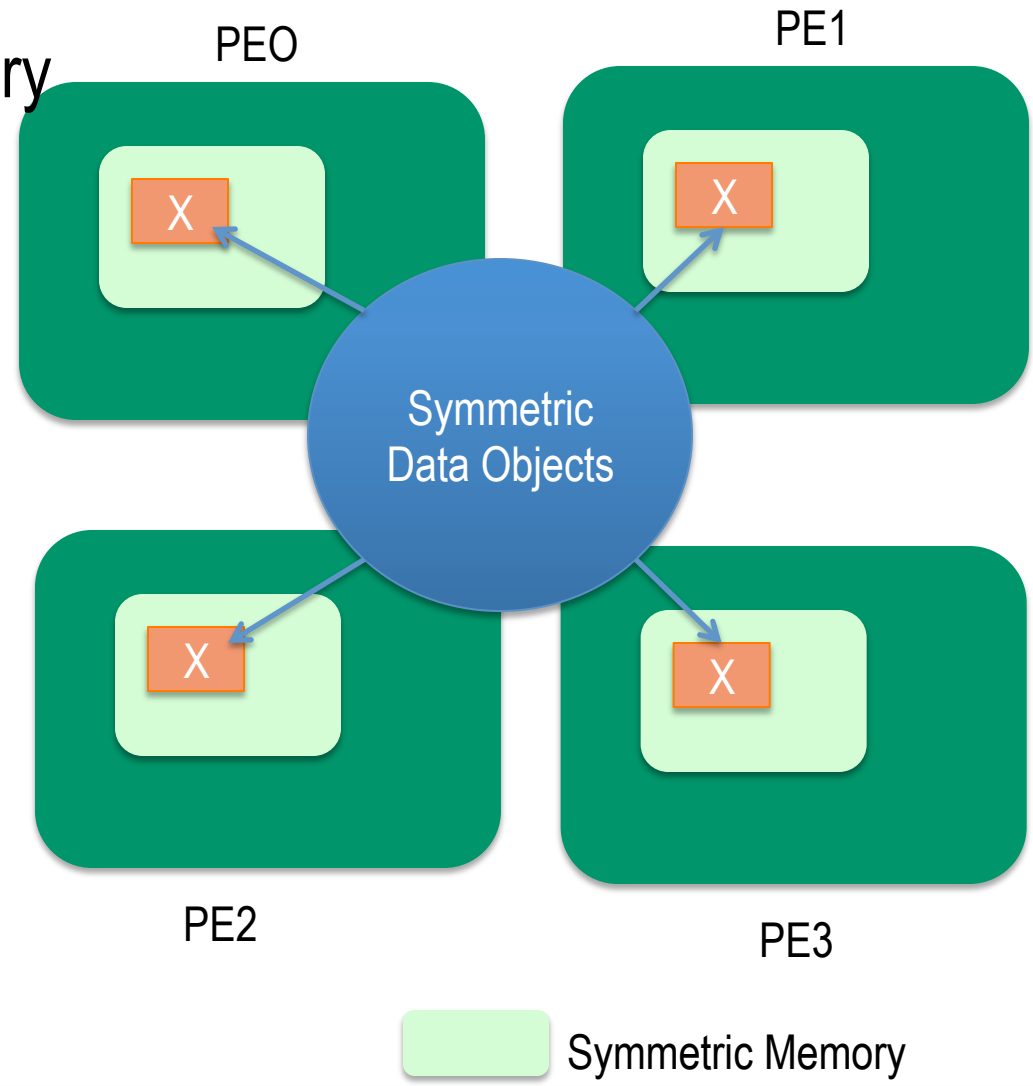
```
int main (void) {
int *x;

start_pes(0);

...
x = (int*) shmalloc(sizeof(x));

…

shmem_barrier_all(); ...
shfree(x);
return 0;
}
```



OpenACC

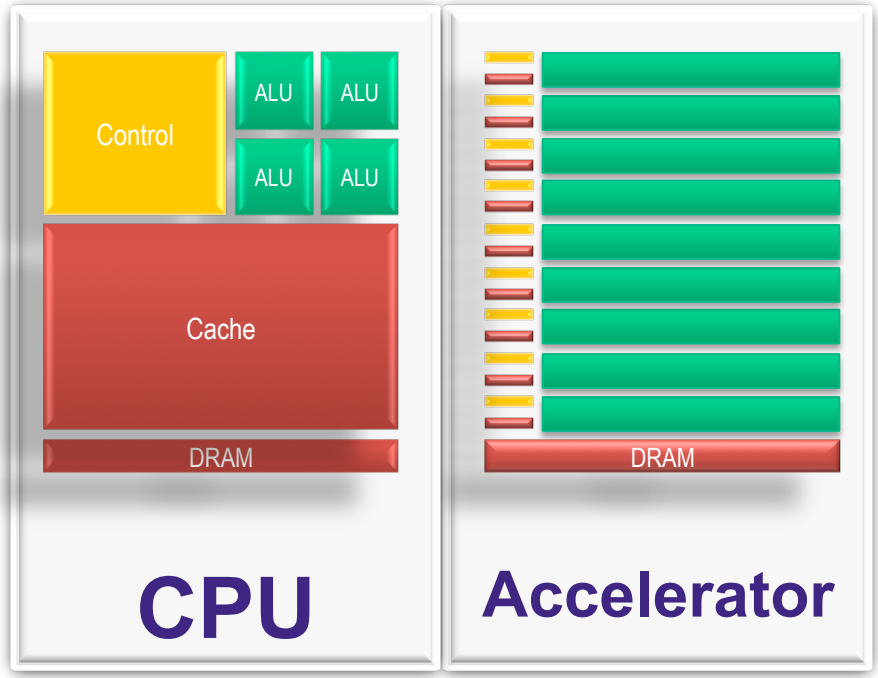
OpenACC
Directives for Accelerators

- A directive-based API to program accelerators (C/C++/FORTRAN)
  - www.openacc.org (current OpenACC 2.0 specification)
- Express data and computations to be executed on an accelerator
  - Incrementally marks accelerated code regions
- Main OpenACC constructs
  - Parallel and kernel regions
  - Parallel loops
  - Data regions
  - Runtime API

OpenACC code snipped

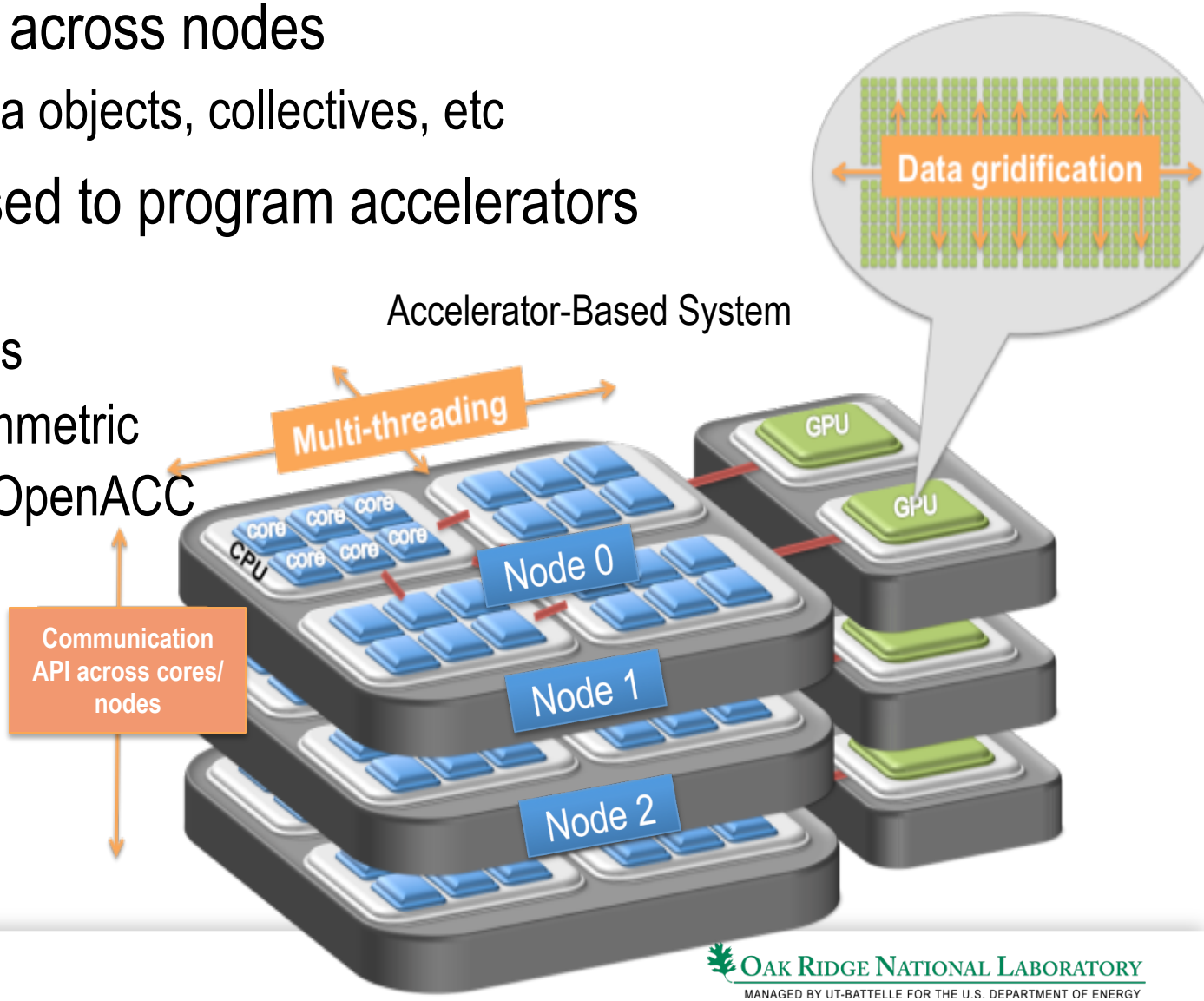
```
#pragma acc kernels
{ ...
#pragma acc loop gang(NB) worker(NT)
  for (int i = 0; i < n; ++i){
    #pragma acc loop vector(NI)
    for (int j = 0; j < m; ++j){
      B[i][j] = i * j * A[i][j];
    }
  }
}
```

Accelerator-based offload model



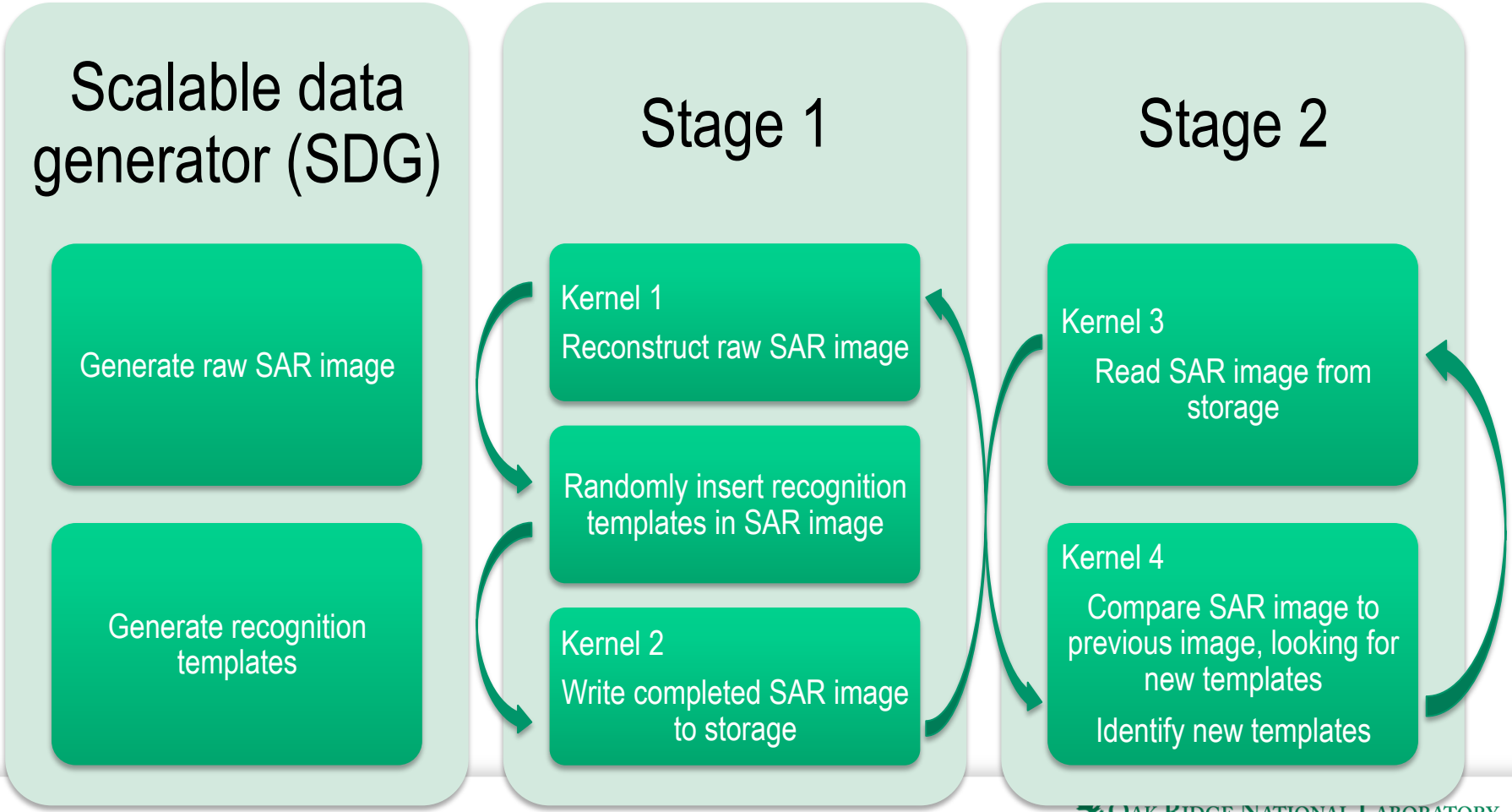
OpenSHMEM / OpenACC hybrid program

- OpenSHMEM is used to program communication across nodes
  - Symmetric data objects, collectives, etc
- OpenACC is used to program accelerators
  - Data regions
  - Parallel regions
- OpenSHMEM symmetric variables used in OpenACC data regions.



Evaluation: SAR image reconstruction

- Benchmark processes Synthetic Aperture RADAR image
- Uses image recognition to detect templates
- Written in C

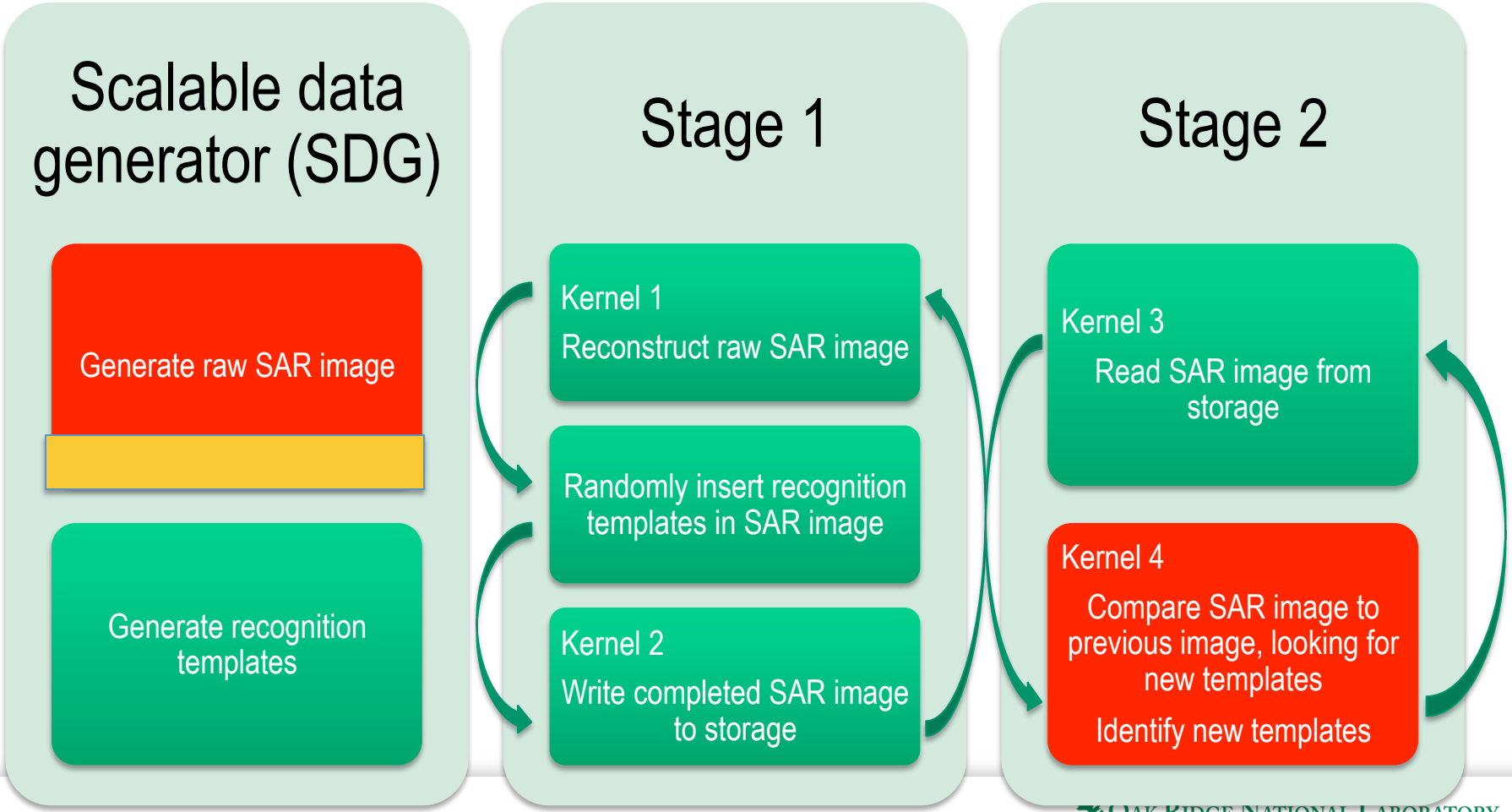


OpenSHMEM / OpenACC

- OpenACC
  - Used for data regions and launch parallel regions in:
  - Raw SAR data construction
    - Very fast at processing image data
  - Image recognition kernel (Kernel 4)
  - For benchmark, can have a simple conditional at runtime that runs same loop either on GPU or CPU
- OpenSHMEM
  - Final reconstruction of raw SAR data
    - Work distributed among multiple GPUs
    - After work is done data is moved to main memory and reduction done among all nodes
  - All statistics counting is done with OpenSHMEM

Evaluation: SAR image reconstruction

- OpenACC to accelerate kernels
- OpenSHMEM to communicate results



SSCA3 on TITAN (TOP500: #2)

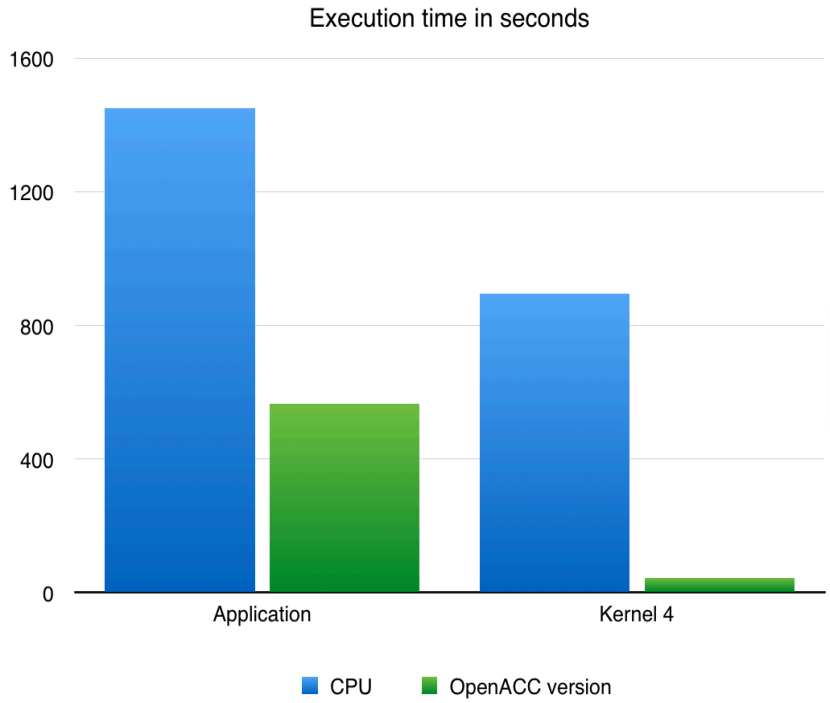
OAK RIDGE National Laboratory

Experimental Setup:

- AMD Opteron 6274, 16 cores 2,2 GHZ
- GPU NVidia Tesla K20x
- capsmc 3.4.2 with Intel 13.1.3

Speed-Up :

- Kernel 4: 20x, 36% accelerated
- Application: 2x, 56% serial



Conclusiosn: OpenSHMEM / OpenACC

- Scalable data generator
  - Raw SAR image data allocated on symmetric heap
  - Much of the image processing done with OpenACC
- OpenSHMEM / OpenACC is a viable model
  - Symmetric memory can be ‘copyin/out’ into accelerator data region
  - All one-sided put/gets should be done outside of accelerator data region
    - Implementations needs to optimize these
- OpenSHMEM / OpenACC needs to work more to interoperate
  - OpenSHMEM should be able to access directly the accelerator memory.

Future Work

- Improve Communication
  - Final SAR image reconstruction on GPU
  - Keep all stats on GPU
  - Map symmetric heap onto GPU
- File I/O
  - Do network I/O from GPU or via OpenSHMEM?
  - Local I/O too?
- Incorporate Accelerator-based Libraries
  - Code uses FFT routines
  - No especially good way to leverage CFFT
    - Possible, but not not pretty

Acknowledgements

This work was supported by the United States Department of Defense & used resources of the Extreme Scale Systems Center at Oak Ridge National Laboratory.

Questions?