# Extreme Heterogeneity for $S_n$ Transport Codes

Sunita Chandrasekaran, Assistant Professor, University of Delaware, schandra@udel.edu
(Collaborators: Oscar Hernandez, Wayne Joubert, Oak Ridge National Lab)

Architectures are rapidly evolving, and the future machines are expected to be equipped with large number of different types of devices offering billion-way concurrency. Such extreme heterogeneity demands that we rethink algorithms, languages, programming models among other components in order to increase parallelism from a programming standpoint in order to be able to migrate large scale applications to these massively powerful platforms. Although directive-based programming models is an effective solution as it allows programmers to worry less about programming and more about science, expressing complex parallel patterns in these models can be a daunting task especially when the goal is to match the performance that the hardware platforms are ready to offer. One of such complex parallel patterns, is the wavefront-based motif. This particular pattern has posed numerous challenges and been studied about at length since 1986 [1]. In scientific codes, this pattern has been observed in several structured and unstructured applications including $S_n$ radiation transport codes, linear equation solvers, bioinformatics, nuclear reaction, etc.

## Challenges

The primary challenge faced when attempting to develop a domain specific abstraction for wavefronts that can be implemented on the wide variety of hardware, as well as providing enough foresight into hardware trends to allow for adoption on future extreme heterogenous architectures with little to no modifications to the code. An important secondary challenge revolves around maintaining performance-portability while undergoing re-implementation of an abstraction, assuming the primary challenge is already met with a solution.

In the case of wavefront-based parallel patterns, this is extremely difficult. In addition to having a complex parallel pattern with regards to computation, different wavefront codes have non-uniform data dependencies. The commonality between these codes is that all data dependencies are met upstream. In a serial implementation, this implies that all data dependencies for a given iteration are satisfied by a previous iteration. There is no reliance on future computation. However, this becomes a huge problem when implementing a parallel wavefront code because we must structure our problem space in such a way that we expose parallelism, while ensuring that all data dependencies are met. This seems fairly trivial at first glance, but different wavefront codes have different amounts of data dependencies. They are all in the upstream direction, but we have no way of knowing what number of neighboring cells in the upstream direction a particular algorithm will examine. Some algorithms, such as ORNL's Minisweep proxy code for Denovo radiation transport application, examine just a single neighboring cell in each upstream direction. Minisweep was recently parallelized and accelerated on state-of-the-art systems [5]. Others, like Smith-Waterman, trace back through an unknown number of neighboring cells until they find a particular value. This makes incorporating a data representation into a high-level abstraction very tricky.

## Research Direction

***Abstract machine model:*** Modern compute node hardware has an execution hierarchy. For example, a compute node may be composed of multiple GPUs, each with multiple cores possessing hardware threads and employing vector units composed of vector lanes. Some of these have co-located memories, for example node main memory, GPU high bandwidth memory or GPU shared memory associated with a streaming multiprocessor (SM) core. Execution threads are also associated with each level: for NVIDIA GPUs, in-warp threads execute in lock-step within a warp, in-threadblock threads are associated with an SM, and the thread grid is associated with the GPU. One can thus view a node as a hierarchy of execution units, memory places and compute threads, and in particular hardware threads can be thought of as indexed as a tuple depending on the location in the hierarchy. Threads also have characteristics based on location, e.g., thread synchronization across different cores of a node may be impossible or much slower compared to on-core synchronization. Likewise, memories at different levels have different speeds, and thread access to memories may have NUMA effects depending on the level. Note that these concepts readily apply to heterogeneous node as well as homogeneous systems.

**State-of-the-art Research** Wavefront parallelization approaches have been mapped to CPUs, GPUs, FPGAs Cell BEs. These approaches include blocking, loop permutation and skewing implemented within a polyhedral compiler transformation framework, adopting Lamport's original parallel pipelined wavefront 'hyperplane' algorithm for certain applications [2-4]. Other approaches include HPCS languages, preliminary studies that use TBB, Cilk, CnC. In spite of these efforts, it is clear that most of these strategies either do not solve the wavefront parallel pattern itself but offer solutions to a specific similar problem type or require the user to use a low-level programming language to create loop transformations incurring a steep learning curve or provide a hardware-specific solution. It is hard and almost impossible for a scientific code developer to take these solutions for his/her own code and target modern platforms. Such gaps in the state-of- the-art work serve as a motivation for us to rethink what we need to support such computational motifs on extreme heterogeneous systems.

**Maturity:** The above state-of-the-art summary indicates that in spite of existing efforts, they cannot be leveraged by scientific developers for their applications. We propose to create an abstract parallelism model and create language extensions at the high-level to tackle this problem.

**Timeliness:** Architectures are becomingly increasingly parallel with fat and thin cores along with several tiers of memory hierarchy. Low-level programming model may achieve the best performance but expecting a domain scientist to be an expert low-level programmer to tap into the systems' potential may not happen.

**Uniqueness:** There are several computational motifs that exposes wavefront-based pattern but currently there is no high-level solution to map such a pattern to extreme heterogeneity system. It is timely to create a high-level language extension with an abstract parallel model in mind to tackle this problem.

**Novelty:** The state-of-the-art paragraph captures existing methods that tackles this problem. However, those solutions cannot be adopted by scientific applications, as they are either too low-level or tied to a specific framework that is not being used by these applications. Moreover, it is not humanely possible to update the code every time the hardware changes. There is no work to the best of our knowledge that is exploring this approach. We aim to work with programming model standard communities and the vendors, explore a suitable solution that can be used to develop a solution applicable to several computational motifs exposing this parallel pattern.

## REFERENCES

[1] Wolfe, M., 1986. Loops skewing: The wavefront method revisited. International Journal of Parallel Programming, 15(4), pp.279-293.
[2] DOE Accelerated Strategic Computing Initiative, The ASCI Sweep3D Benchmark Code, 1995
[3] Lamport, L., 1974. The parallel execution of DO loops. Communications of the ACM, 17(2), pp.83-93.
[4] Pennycook, S.J., Hammond, S.D., Mudalige, G.R., Wright, S.A. and Jarvis, S.A., 2011. On the acceleration of wavefront applications using distributed many-core architectures. The Computer Journal, 55(2), pp.138-153
[5] R. Searles, S. Chandrasekaran, W. Joubert, O. Hernandez, "MPI + OpenACC: Accelerating Radiation Transport Mini-Application, Minisweep, on Heterogeneous Systems," in Computer Physics Communications, CPC 2018. DOI: 10.1016/j.cpc.2018.10.007