- **Project Information**

    - Reducing jet aircraft noise by harnessing the heterogeneous XK nodes on Blue Waters
    - Professor Daniel J. Bodony, Department of Aerospace Engineering, University of Illinois at Urbana-Champaign
    - Daniel Bodony (bodony@illinois.edu)

- **Executive summary (150 words)**

Jet engine noise caused by military and commercial aircraft is an unwanted byproduct that negatively impacts the quality of life for residents living near airports and creates operational challenges and risks for carrier-borne personnel. Without effective reduced-order models that connect jet engine designs to the noise they create, large scale simulations are required to predict jet engine noise and develop noise-reducing designs. The computational demands of jet noise predictions are beyond current petascale systems so we are developing the tools needed for a computational fluid dynamics code to utilize heterogeneous architectures envisioned for future exascale systems. Our OpenCL-OpenMP-MPI tool has demonstrated performance on 4K XK7 BW nodes.

- **Description of research activities and results**

Key Challenges: Developing quieter jet engines that achieve the same aerodynamic performance (thrust and fuel consumption) requires detailed knowledge of how the engine design impacts the sound-generating motion of the exhaust gases. Unfortunately, no applicable reduced-order theory exists that accurately connects the design to the sound. Instead, experimentation or numerical predictions are required. The fundamental problem that must be overcome is predicting the generation of sound by a compressible, viscous fluid in turbulent motion.

Why it Matters: The computational demands of simulating turbulence required for accurate jet noise predictions exceeds current computational capacity. An MPI-only expression of the parallelism in our computational fluid dynamics (CFD) code *PlasComCM* performs well on existing homogeneous petascale computers, such as Blue Water's XE nodes, but cannot leverage the compute capability contained in the XK nodes nor on the half of the projected CORAL-procured exascale machines under development by the Department of Energy. Machines like Sierra are expected to be comprised of IBM Power 9 CPUs with limited double precision capability that will manage two NVIDIA GPGPUs connected by NVLINK that will perform most of the double precision arithmetic. However, the NVLINK bandwidth is still small compared with that on the main bus and care must be taken to ensure memory coherency as well as minimization of data transfer.

Why Blue Waters: Blue Waters' XK7 nodes are a surrogate for the expected heterogeneity and this project seeks to develop the software tools required to transform an existing MPI-only CFD code into one that take advantage of on-node heterogeneity. The XK7 node's NVIDIA GPGPU is connected to the host via the PCIe bus with limited bandwidth and thus represents the fundamental exascale challenge in managing task parallelism on a node with asymmetric computing capability, asymmetric memory hierarchies, and asymmetric memory bandwidth.

Accomplishments: Over the past year, our CFD code has been refactored to utilize an MPI + OpenMP + X model to handle compute asymmetry and GPGPU management. At the node level, one or more MPI processes is launched and in charge of managing a local section of the global CFD problem to be solved. Each MPI process uses the C++ futures feature to launch two additional threads: one thread manages computation on the host socket using OpenMP parallelism and the other thread manages computation on the device using OpenCL for the data management. Inter-MPI communication is handled using standard message passing.

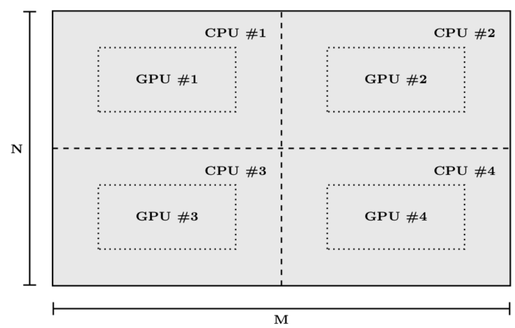

Figure 1. Halo data partitioning for on-node parallelism.

The local problem is partitioned in a "halo" inspired way (see figure 1) where the portioned owned by the GPGPU is in the interior to the local problem owned by the host. In doing so the inter-GPU communication need is avoided, which has large latency due to distance and software layers, but at the expense of sending more data locally. The data owned by the CPU is computed on in a series of independent tasks while the data sent to the GPGPU is reordered to ensure coalesced memory accesses.

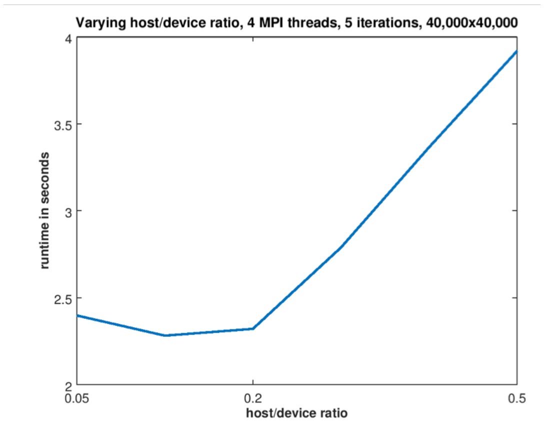

Figure 2. Evaluation of optimum host-to-device work distribution on the XK7 node.

The most important parameter is the ratio of work sent to the GPGPU and that retained on the host CPU. For our test problem of red-black successive over relaxation, which has a low compute intensity, the optimum distribution has roughly 10% of the work on host and 90% of the work on the device. At higher computational intensities we expect more work to be sent the CPU to be optimal; see figure 2. The corresponding measured speed up is on the order 2-3 times over the CPU-only version as shown in figure 3.

- **List of publications and presentations associated with this work**

  ECSS Experience: Performance of a CFD Code Running on Stampede's Intel Xeon Phi in Symmetric Mode, by W. Zhang, D. J. Bodony, J. Larson, and L. Wilson, poster presented at XSEDE'15 in St. Louis, Missouri.

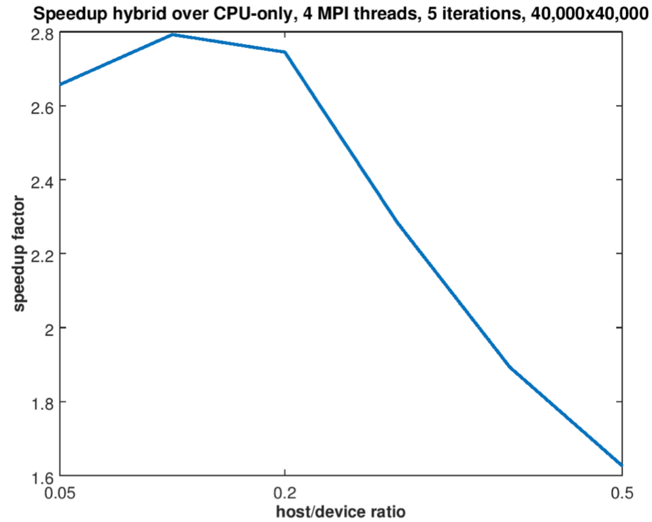

Figure 3. Measured program speedup on XK7 nodes using MPI + OpenMP + OpenCL

- **Plan for next year**

  The RBSOR tests conducted at scale on Blue Waters's XK partiation need to be replicated in our production CFD tool. We are currently merging the two approaches and testing runs now. Our performance measurements will include single-node performance, as measured by percentage of peak based on a performance model for the node, as well as cross XK7-node tests using all of the available XK7 partition. In Q1 we will have a working tool for which production-level runs will be conducted to measure performance as well as to predict jet noise and will continue to use that tool throughout Q2—Q3. In Q4 we anticipate refined performance measurement and code optimization. The tool will be tested on jet noise prediction simulations for which we anticipate needing 400,000 XK nodes hours for 2017-2018 and will split that between the four quarters as Q1: 30%, Q2: 30%, Q3: 30%, Q4: 10%. Our storage needs will be 500 TB for the entire year.