Direct numerical simulations of reacting flows with detailed chemistry using many-core/GPU acceleration


*Clean Combustion Research Center, King Abdullah University of Science and Technology (KAUST), Thuwal 23955–6900, Saudi Arabia
*Gwangju Institute of Science and Technology, Gwangju 61005, Korea
*Oak Ridge National Laboratory, Oak Ridge, TN 37831–6008, USA

Abstract

A new direct numerical simulation (DNS) code for multi-component gaseous reacting flows has been developed at KAUST, with the state-of-the-art programming model for next generation high performance computing platforms. The code, named KAUST Adaptive Reacting Flows Solver (KARFS), employs the MPI+X programming, and relies on Kokkos for “X” for performance portability to multi-core, many-core and GPUs, providing innovative software development while maintaining backward compatibility with established parallel models and legacy code. The capability and potential of KARFS to perform DNS of reacting flows with large, detailed reaction mechanisms is demonstrated with various model problems involving ignition and turbulent flame propagations with varying degrees of chemical complexities.

Keywords: Direct numerical simulation, combustion, chemistry solver, GPU acceleration, MPI+X programming

*Corresponding author.
E-mail address: hong.im@kaust.edu.sa (Hong G. Im)

Preprint submitted to Computers and Fluids
1. Introduction

Advances in high performance computing (HPC) technology enable high fidelity computational fluid dynamics (CFD) simulations of reactive flows at realistic device scales and parametric conditions. While there exists a number of reduced-order descriptions of such complex flows, direct numerical simulation (DNS) is considered a first-principle approach for turbulent combustion simulations without statistical averaging or spatial filtering. The conservation equations describing turbulent reactive flows solve for a large number of solution variables including mass, momentum, energy, and chemical species concentrations. The large system of partial differential equations is solved with full spatial and temporal resolution using discretization schemes with minimal numerical dissipation, since accuracy is of primary importance. Today’s detailed simulations for combustion of hydrocarbon fuels typically involve tens of species and hundreds of elementary kinetic reactions, thus significantly increasing the computational cost. Moreover, turbulent reacting flows span a large spectrum of length and time scales, demanding a substantial number of grid points and time steps in order to capture all relevant physical scales. With modern power generation and propulsion systems pushing the operating ranges to extreme conditions at higher pressure and turbulence intensity, DNS of turbulent combustion of realistic fuels at relevant scales remain a significant challenge.

In recent years, the HPC systems have seen the introduction of accelerators based on heterogeneous architectures utilizing many-core and graphics processing unit (GPU) processors. The newer computing hardware framework is markedly different from the earlier generation of single- and multi-core processors that relied mainly on the clock speeds to achieve performance growth in floating point operations per unit time. In contrast, next generation supercomputers are composed of a large number of light-weight cores with wide vector instructions. HPC installations and scientific computing user facilities around the world have been increasingly adopting these new architectures for several reasons such as cost, performance, power consumption, cooling, and space. Scientific simulation codes can potentially achieve dramatic speed-ups on the modern HPC systems based on many-core and GPU architectures. Such speed-ups have the potential to shift the use of CFD to much more complex problems, with greater precision and within acceptable solu-
tion times. However, significant challenges remain in achieving these potential benefits, including the ability to adapt the numerical algorithms and port the physics kernels to a programming model that is conducive to achieving the optimal performance on these architectures.

The most common approach used by CFD groups porting their code to the modern architectures is to select the most time-consuming and compute-intensive kernels. These kernels are then optimized with multi-threading for many-core architectures or offloaded to the GPUs for accelerated solution [1]. Such a *kernel-based* approach, however, cannot achieve speed-ups above a certain limit. The fraction of the computational work that is programmable as kernels sets the upper limit that can be offloaded to the target architectures. More often, CFD codes have a core flow solver that consumes a significant fraction of the computational resources. As such, the speed-ups cannot be improved unless the core solvers are ported. Main CFD codes in the fluid dynamic and combustion community were developed during an era when the traditional CPU architectures with deep instruction pipelines and the increase in clock frequency were the prime factors behind the HPC performance improvements. Barring a few instances, such as Ref. [2], it is generally difficult to modify software written for conventional processor architectures to obtain good performance on the modern processor architectures. GPUs and many-core processors require particular attention to some performance-critical aspects of the software such as data locality, layout and how well the parallelism is exposed. At times, a complete rewrite of the software in a newer programming model is warranted and a more productive path forward.

In contrast to programming models that require a rewrite of the software, directives based approaches, such as OpenACC and OpenMP, have been developed for accelerating legacy codes. In these approaches, various directives are used to expose the underlying data structures and hierarchy of parallelism. A compiler that supports these directives can potentially generate an efficient code for the multiple target architectures. Levesque et al. [3] found that efficient acceleration of the code through directives requires refactoring of the data structures and rearrangement of loops to achieve higher granularity in the loop regions. Their experience has also shown that, while the directives approach emphasizes reuse of the legacy code, a considerable refactoring of the original code is necessary to achieve the appropriate data layout and loop granularity. The present study explores the path of creating a new implementation of our combustion DNS code, named
the KAUST Adaptive Reacting Flows Solver (KARFS) [4, 5], building on the experience with
the numerical methods and physics models used in legacy implementations of structured-grid and
finite difference codes.

In a fresh implementation of the DNS code, modern C++ was adopted as the programming lan-
guage which is predominantly used for innovations in the scientific computing community. Many
of the newer scientific libraries and parallel programming frameworks are being developed in and
for C++. Scientific application developers may also choose from a multitude of programming
models, some of which are architecture-specific. For example, CUDA is a programming model
developed for the NVIDIA GPUs, while Pthreads are geared towards x86 based multi-core and
many-core processors. Such architecture-specific programming models severely limit the porta-
bility of the software and require the developers to maintain multiple implementations if portability
is desired. Programming models, such as OpenCL, that promised to be portable across multiple
vendors and architectures have not been adopted widely and also suffer from lack of performance
portability. Therefore, the present project chose the performance portable programming model
provided by the Kokkos C++ library [6, 7] for implementing the DNS code.

The Kokkos C++ programming model provides the foundation for abstracting the data loca-
tion, layout and parallel execution. Software modules and kernels programmed in Kokkos are
implemented with an additional layer of parallelism separate from the distributed memory parallel
model. In this case, the message passing interface (MPI) is used as our distributed memory par-
allel model for parallelism across networked nodes in a HPC system. The Kokkos programming
model provides on-node parallelism through threads and vectorization. The computational loops
and modules are written using Kokkos parallel patterns to express the work at the finest granular-
ity, in this case the calculations that occur with the grid-level and equation-level parallelism. A
parallel work dispatch pattern is used to generate parallel code over all the finite difference grid
nodes that belong to a single instance of the program’s distributed memory parallel execution. The
parallel work dispatch method used by Kokkos is not much different from the corresponding ap-
proaches used by other C++ based programming libraries such as the Standard Template Library
(STL), Intel’s Thread Building Blocks (TBB) or Thrust for CUDA. However, Kokkos also pro-
vides an abstraction for the data location and layout through its memory spaces and arrays, which
allows the computations to be expressed using multidimensional arrays that are not dissimilar to the Fortran array syntax.

The main scope of this paper is to introduce our new in-house HPC reacting flow DNS solver, KARFS, as a showcase to implement the Kokkos programming model towards accelerated performance in modern supercomputing hardware. In the following, the conservation equations and computational algorithms employed in KARFS are introduced. Subsequently, various types of parallelism are implemented for different code modules and the resulting levels of acceleration are examined. Demonstrations of fidelity and improved performances will be shown in one-dimensional and multi-dimensional physical problems commonly studied in combustion research.

2. Governing equations

The newly developed KAUST Adaptive Reacting Flow Solver [4, 5] is a high-fidelity scientific code for direct numerical simulations of combustion with detailed chemistry and transport. The code describes the gaseous combustion problems in fully compressible formulation. For a gaseous mixture of $N$ species evolving in time ($t$) and space ($x_j$), the equations for the conservation of species, mass, momentum, and energy are written as:

$$\frac{\partial}{\partial t} (\rho Y_i) + \frac{\partial}{\partial x_j} \left[ \rho Y_i (u_j + V_{i,j}) \right] = W_i \dot{\omega}_i, \quad i = 1, \ldots, N \quad (1)$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j} (\rho u_j) = 0, \quad (2)$$

$$\frac{\partial}{\partial t} (\rho u_i) + \frac{\partial}{\partial x_j} (\rho u_i u_j) + \frac{\partial p}{\partial x_j} = \frac{\partial \tau_{ij}}{\partial x_j}, \quad l = 1, 2, 3 \quad (3)$$

$$\frac{\partial}{\partial t} (\rho E_t) + \frac{\partial}{\partial x_j} \left[ (\rho E_t + p) u_j \right] = \frac{\partial}{\partial x_j} (u_j \tau_{ij}) - \frac{\partial d_j}{\partial x_j}, \quad (4)$$

where $\rho$ is the mixture density, $Y_i$ is the mass fraction of species $i$, $u_j$ is the flow velocity component in the $j$th direction, $V_{i,j}$ is the diffusion velocity of species $i$ in the $j$th direction, $W_i$ is the molar mass of species $i$, $\dot{\omega}_i$ is the reaction rate of species $i$, $p$ is the mixture pressure, $\tau_{ij}$ is the viscous
stress tensor, $E_t$ is the mixture specific total energy, and $q_j$ is heat flux vector in the $j$th direction. The specific total energy is defined as:

$$E_t = \frac{u_j u_j}{2} + \sum_{i=1}^{N} (h_i Y_i) - \frac{p}{\rho},$$  

(5)

$$h_i = h_i^0 + \int_{T_0}^{T} c_{p,i} \, dT,$$  

(6)

with $h_i$ being the enthalpy of species $i$, $T$ the mixture temperature, $h_i^0$ the enthalpy of species $i$ at reference temperature $T_0$, and $c_{p,i}$ the constant-pressure heat capacity of species $i$. The ideal gas equation of state is used as a constitutive relation

$$p = \rho R T,$$  

(7)

where $R$ is the gas constant, determined in terms of the universal gas constant, $R_u$, and mixture composition as $R = R_u \sum_{i=1}^{N} Y_i / W_i$. The diffusion velocity $V_{i,j}$ and the heat flux vector $q_j$ are described as

$$V_{i,j} = -D_i \frac{1}{Y_i} \frac{\partial Y_i}{\partial x_j},$$  

(8)

$$q_j = -\lambda \frac{\partial T}{\partial x_j} + \rho \sum_{i=1}^{N} h_i Y_i V_{i,j},$$  

(9)

where $D_i$ is the mass diffusivity of species $i$ into the mixture and $\lambda$ is the mixture thermal conductivity, while the viscous stress tensor is given by

$$\tau_{lj} = \mu \left( \frac{\partial u_l}{\partial x_j} + \frac{\partial u_j}{\partial x_l} \right) - \frac{2}{3} \mu \delta_{lj} \frac{\partial u_l}{\partial x_l},$$  

(10)

with $\mu$ and $\delta_{lj}$ being the mixture viscosity and Kronecker delta, respectively. The reaction rate $\dot{\omega}_i$ is written as

$$\dot{\omega}_i = \sum_{k=1}^{M} (\nu_i^{',k} - \nu_i^{',k}) \left( K_{f,k} \prod_{n=1}^{N} C_n^{\nu_n^{',k}} - K_{b,k} \prod_{n=1}^{N} C_n^{\nu_n^{',k}} \right) C_{M,k},$$  

(11)

where $\nu_i^{',k}$ and $\nu_i^{',k}$ are the forward and backward reaction coefficients of species $i$ for elementary reactions $k = 1, \ldots, M$. Likewise, $K_{f,k}$ and $K_{b,k}$ are the forward and backward rate constants for the elementary reaction $k$. The third body exponent $\alpha_k$ is 1 if the reaction $k$ involves a third body and
0 otherwise. The molar concentrations of species $n$ and of a third body associated with reaction $k$ are obtained using the molar mass $W_n$, mass fraction $Y_n$, and third-body efficiency $\eta_{n,k}$ as

$$C_n = \frac{\rho Y_n}{W_n},$$

(12)

$$C_{M,k} = \sum_{n=1}^{N} \eta_{n,k} C_n = \sum_{n=1}^{N} \eta_{n,k} \frac{\rho Y_n}{W_n}.$$  

(13)

3. Numerical method

KARFS solves the compressible Navier–Stokes, species and energy equations fully explicitly employing a fourth-order explicit Runge–Kutta method for time integration and an eighth-order, non-dissipative, central differencing scheme for spatial discretization [8]. Spurious high wavenumber contents developed due to the energy cascade into sub-grid scales are filtered out using a tenth-order dissipative filter. Moreover, the Navier–Stokes characteristic boundary conditions (NSCBC) [9] for reacting flows have been implemented.

KARFS has performance-portable capabilities for multi- and many-core heterogeneous platforms, which is instrumental to meet demands of exascale computing machines. The employed Kokkos framework allows KARFS to straightforwardly utilize the same source code on CPUs and GPUs while maintaining performance comparable to the code that is optimally tuned to either processing unit [10]. In addition, KARFS utilizes the Cantera [11] library to incorporate thermodynamic, transport and chemical kinetic data into computation. While Cantera has a full capability to compute thermodynamic and transport properties as well as reaction rates, independent performance-critical kernels were also implemented for enhanced performance and portability to GPU accelerators.

4. Results and discussion

4.1. KARFS MPI+X hierarchical parallelism

First, the KARFS GPU acceleration and portability were tested using a set of two-dimensional (2-D) auto-ignition problems with computational grid sizes ranging from $512^2$ up to $3,072^2$ points.
These benchmark runs were conducted on a developmental system at KAUST consisting of two Intel Haswell CPUs (16 cores each) and eight NVIDIA Tesla K80 accelerators with a dual-GPU design. This testing machine has the same CPU specifications as the computational nodes of the Shaheen II supercomputer at KAUST [12], which is a Cray XC40 system delivering over 7.2 PFLOPS of theoretical peak performance and having a total of 197,568 CPU cores running at 2.3 GHz and 790 TB of aggregate memory. Unless otherwise stated, the time integration of the governing equations was carried out with the explicit Runge–Kutta scheme.

A reduced kinetic mechanism for dimethyl-ether (DME) consisting of 30 species was used to represent chemical reactions [13]. This reduced mechanism was adapted so that the reaction rates could also be evaluated on GPUs. A number of combinations for the MPI+X hierarchical parallelism of KARFS, where X stands for OpenMP and/or CUDA backends, were tested and the respective speed-up was assessed. Figure 1 illustrates the KARFS speed-up against a single core performance for different CPU/GPU combinations, using one CPU and up to 8 GPUs. The maximum achieved speed-up was over 500 times against a serial run on a single CPU core, with the combination of one CPU (partitioned for 8 MPI domain decomposition with 2 OpenMP threads) and 8 GPUs (one GPU per each MPI domain). GPU acceleration enabled a speed-up of about 40 on top of the full utilization of CPU cores.

4.2. Acceleration for detailed and large reaction mechanisms

Detailed chemical reaction mechanisms developed for practical fuels vary in size from tens to thousands of species and hundreds to thousands of reactions, often consisting of an enormous spectrum of time scales leading to severe stiffness. As an attempt to demonstrate the code performance with large degrees of freedom, more complex reaction mechanisms were employed and two approaches for performing kinetic computations with the GPU were evaluated. In one approach, a library for generic evaluation of reaction rates that uses CUDA kernels was employed [14] (GPU-A approach). In the other, for a given reaction mechanism CUDA kernels for the evaluation of reaction rates were explicitly written with the help of pyJac [15], a tool to compute analytical Jacobians (GPU-B approach).

First, a reaction mechanism for DME consisting of 39 species and 175 reactions [13] was
considered. The same mixture as in the above test was used, although different two- and three-
dimensional grids were considered, ranging in size from $96^2$ to $48^3$ points. The computational
costs and speed-ups associated with each case and approach are summarized in Table 1. The use
of the second approach resulted in faster computations and, for the case with the largest number
of grid points, a larger acceleration of about two was obtained. However, for the smallest problem
size, no acceleration was attained. This confirms a well-known fact that for a small size problem
the use of GPU does not benefit computational efficiency due to the overhead of transferring data
between the host CPU and the GPU.

Next, a significantly larger kinetics scheme for the oxidation of n-heptane (654 species, 5,258
reactions) [16] was considered. These types of fuel and reaction mechanism are more represen-
tative of complex hydrocarbon fuels. Although a similar benchmark was conducted, results for
the approach GPU-A are reported only, since this can be readily applied in a general fashion to
any reaction mechanism and is expected to be more widely employed within our DNS framework.
The computational costs and speed-ups associated with this case are listed in Table 2. For the
largest grid, a speed-up of 11.3 was attained when using the GPU. In contrast to the previous case
for DME (39 species and 175 reactions), a noticeable level of acceleration was found even for the smallest grid because the overall increase in the problem size makes an effective use of the GPU computation. Considering that future HPC platforms such as Summit at ORNL are expected to have multiple Volta GPUs (3) per CPU, interconnected using NVLink, the present test results demonstrate that DNS of reacting flows with an unprecedented scale of complex and detailed chemistry will indeed be possible.

<table>
<thead>
<tr>
<th>Grid size</th>
<th>1 CPU (s/step)</th>
<th>GPU-A (s/step)</th>
<th>GPU-B (s/step)</th>
<th>Speed-up (GPU-B vs. CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>96(^2)</td>
<td>0.699</td>
<td>1.377</td>
<td>1.289</td>
<td>0.54</td>
</tr>
<tr>
<td>32(^3)</td>
<td>2.206</td>
<td>2.073</td>
<td>1.801</td>
<td>1.23</td>
</tr>
<tr>
<td>48(^3)</td>
<td>7.046</td>
<td>3.903</td>
<td>3.489</td>
<td>2.02</td>
</tr>
</tbody>
</table>

Table 1: Cases, cost of time-step evaluation and speed-up for larger DME mechanism (39 species, 175 reactions) using two different approaches for the computation of chemistry on the GPU. (1 CPU equals 16 cores).

<table>
<thead>
<tr>
<th>Grid size</th>
<th>1 CPU (s/step)</th>
<th>1 GPU (s/step)</th>
<th>Dual GPU (2 MPI ranks) (s/step)</th>
<th>Speed-up (Dual GPU vs. 1 CPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>96(^2)</td>
<td>52.389</td>
<td>25.937</td>
<td>22.368</td>
<td>2.3</td>
</tr>
<tr>
<td>32(^3)</td>
<td>179.91</td>
<td>37.189</td>
<td>29.192</td>
<td>6.2</td>
</tr>
<tr>
<td>48(^3)</td>
<td>598.481</td>
<td>81.09</td>
<td>52.847</td>
<td>11.3</td>
</tr>
</tbody>
</table>

Table 2: Cases, cost of time-step evaluation and speed-up for n-heptane mechanism (654 species, 5,258 reactions) using the GPU-A approach for the computation of chemistry on the GPU. (1 CPU equals 16 cores and a dual GPU corresponds to one K80 accelerator).

4.3. **GPU acceleration of stiff-ODE solver and Cantera**

Complex chemistry often involves a wide spectrum of time scales and requires effective strategies to deal with the numerical stiffness. In this regard, KARFS also has the capability to implicitly integrate chemical kinetics on GPU using a Cantera-CVODE-MAGMA framework. CVODE is an integrator for stiff ordinary differential equations (ODE), which is part of Sundials [17].
MAGMA [18] is a library for linear algebra on hybrid architectures, and has been linked to CVODE and further coupled with Cantera. By coupling CVODE with MAGMA, the linear algebra computations associated with the stiff-ODE integrator are accelerated. The Cantera-CVODE-MAGMA framework, without modification of Cantera’s original reaction-rate evaluation functionality, was utilized to compute ignition delay times (zero-dimensional (0-D) homogeneous problem), with various reaction mechanisms including one of the largest available in the community, consisting of 7,172 species and 47,157 reactions [19]. Table 3 shows a comparison among calculations using a regular chemical kinetics solver of Cantera, the GPU-accelerated solver using a single CPU core plus a GPU, and one making use of the Intel MKL library for threading with 16 CPU cores. It is shown that for very large chemical kinetics mechanisms, a speed-up of two orders of magnitude is achievable with accelerated solvers compared to the standard solver running on a single CPU core.

<table>
<thead>
<tr>
<th>Mechanism size</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of species</td>
<td>53</td>
<td>2192</td>
<td>7172</td>
</tr>
<tr>
<td>Number of reactions</td>
<td>325</td>
<td>13927</td>
<td>47157</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Wall clock (s)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sundials CPU 1</td>
<td>0.23</td>
<td>2712</td>
<td>91583</td>
</tr>
<tr>
<td>Sundials-MKL CPU 16</td>
<td>0.14</td>
<td>41</td>
<td>722</td>
</tr>
<tr>
<td>Sundials-MAGMA GPU 1</td>
<td>2.0</td>
<td>43</td>
<td>343</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Speed-up</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Sundials CPU 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sundials-MKL CPU 16</td>
<td>1.6</td>
<td>66</td>
<td>127</td>
</tr>
<tr>
<td>Sundials-MAGMA GPU 1</td>
<td>0.1</td>
<td>63</td>
<td>267</td>
</tr>
</tbody>
</table>

Table 3: Cantera-CVODE-MAGMA acceleration of chemical kinetics integration for representative small, medium and large reactions mechanisms.
4.4. Validation of KARFS: constant-volume auto-ignition

Computations of ignition delays for different fuel and mixtures at various conditions have been conducted and compared against results obtained with the standard version of Cantera [11], which is widely utilized for canonical problems involving chemical kinetics, thermodynamics, and/or transport processes and has been extensively validated.

Three fuels were considered: dimethyl-ether (DME), syngas (CO/H₂ at a molar ratio of 1:1), and hydrogen (H₂). For each fuel, lean fuel–air mixtures were considered. In terms of chemical kinetics, the mechanisms by Bhagatwala et al. [13] (30 species), Li et al. [20] (12 species, 33 reactions), and Burke et al. [21] (9 species, 23 reactions) were used to describe the oxidation of DME, syngas, and hydrogen, respectively.

Results for three 0-D cases (one for each fuel) are presented. For all cases, the pressure was set to 10 atm. For the DME and hydrogen cases, an equivalence ratio of 0.5 and an initial mixture temperature of 1075 K were selected, whereas for the syngas case an equivalence ratio of 0.6 and an initial mixture temperature of 1100 K were selected, which are slightly higher. The temporal evolution of temperature for three 0-D computations is shown in Figure 2, comparing the predictions of KARFS against solutions obtained with Cantera. It is evident that the solutions computed with KARFS are in excellent agreement with those given by Cantera. These results serve as numerical validation for accurate chemical kinetics integration and prediction of ignition delays by KARFS.

4.5. Validation of KARFS: one-dimensional steady premixed flame

A prototype problem of combustion is the one-dimensional (1-D), planar, steady premixed flame. This canonical configuration was also used for validation purposes. A premixed mixture of H₂–air with an equivalence ratio of 0.7, fresh gas temperature of 300 K and atmospheric pressure was selected. A 1-D domain of 2 cm, discretized with 2,000 grid points, was used for the computational setup and the reaction mechanism of Burke et al. [21] was utilized.

Figure 3 shows the solution obtained with KARFS compared against the one given by the standard Cantera, where the temperature, H₂ mass fraction and H mass fraction profiles are plotted. The KARFS solution agrees well with that of Cantera. Furthermore, an important quantity of
interest is the laminar flame speed, for which KARFS and Cantera yielded 1.356 m/s and 1.342 m/s, respectively, with a relative difference of about 1%.

4.6. Application to three-dimensional turbulent premixed flame propagating in a channel

As a final demonstration of application of KARFS to multidimensional reacting flows, the DNS computation of an initially planar flame propagating into isotropic turbulence was conducted. A three-dimensional simulation was performed for a rectangular channel with an inlet and an outlet along the axis of propagation, while other directions are periodic. A premixed mixture of H₂–air, at an equivalence ratio of 0.7, fresh gas temperature of 300 K and atmospheric pressure, was considered with the description of chemical kinetics by the mechanism of Burke et al. [21].

The turbulence characteristics are such that the Karlovitz number, which is the ratio of the
flame time to the Kolmogorov eddy turnover time, is equal to 128 and the root-mean-square velocity is 18.9 m/s. According to the premixed regime diagram, referred to as the Borghi diagram [22], these conditions fall into the distributed regime of combustion.

The channel has a length of 0.832 cm, with equal width and height of 0.208 cm, and was discretized with 131,072,000 points (1,280 by 320 by 320). This simulation was run for a physical time representing more than 10 eddy turnover times on the Shaheen II supercomputer, utilizing 16,384 CPU cores. Figure 4 shows an instantaneous snapshot of the solution field, illustrating the flame interaction with forced turbulence. A vorticity iso-surface, identifying vortical structures, is shown in green, while an iso-therm of 1200 K, identifying the flame front, is shown in red. The bottom and side planes are colored by H$_2$O$_2$ mass fraction. As can be seen from the figure, the flame surface is highly wrinkled and distorted, and the H$_2$O$_2$ contours reveal a distributed flame

Figure 3: Comparison of spatial distributions for one-dimensional premixed flame between KARFS and Cantera: (a) temperature, (b) H$_2$ mass fraction and (c) H mass fraction.
5. Conclusions

A new in-house HPC reacting flow DNS solver, KARFS, has been introduced as a showcase to implement the Kokkos programming model towards accelerated performance in modern supercomputing hardware. The MPI+X programming model for KARFS relies on Kokkos for “X” for performance portability to multi-core, many-core and GPUs and provides the ability to innovate software development while maintaining backward compatibility with established parallel models and legacy codes. Using the MPI+X programming model, the capability and potential of KARFS to perform DNS of reacting flows with large, detailed reaction mechanisms was demonstrated. Implementation of an efficient stiff-ODE solver and a splitting algorithm following the MPI+X model is underway, which will add robustness and performance gains when dealing with stiff and complex chemistry.
Acknowledgments

The research work was sponsored by King Abdullah University of Science and Technology (KAUST) and made use of the computer clusters at KAUST Supercomputing Laboratory (KSL), and resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Bok Jik Lee was partly supported by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Science and ICT (2017R1A2B4003327). The authors also thank Dr. Hatem Ltaief at KSL for his assistance with the MAGMA library.

References


[12] https://www.hpc.kaust.edu.sa/content/shaheen-ii.


