Contents lists available at ScienceDirect

Combustion and Flame

journal homepage: www.elsevier.com/locate/combustflame

Using SIMD and SIMT vectorization to evaluate sparse chemical kinetic Jacobian matrices and thermochemical source terms

Nicholas J. Curtis[a,*], Kyle E. Niemeyer[b], Chih-Jen Sung[a]

[a] Department of Mechanical Engineering, University of Connecticut, Storrs, CT 06269, USA
[b] School of Mechanical, Industrial, and Manufacturing Engineering, Oregon State University, Corvallis, OR 97331, USA

A R T I C L E   I N F O

Article history:
Received 22 March 2018
Revised 4 September 2018
Accepted 5 September 2018

Keywords:
Chemical kinetics
SIMD
SIMT
Sparse
Jacobian

A B S T R A C T

Accurately predicting key combustion phenomena in reactive-flow simulations, e.g., lean blow-out, extinction/ignition limits and pollutant formation, necessitates the use of detailed chemical kinetics. The large size and high levels of numerical stiffness typically present in chemical kinetic models relevant to transportation/power-generation applications make the efficient evaluation/factorization of the chemical kinetic Jacobian and thermochemical source-terms critical to the performance of reactive-flow codes. Here we investigate the performance of vectorized evaluation of constant-pressure/volume thermochemical source-term and sparse/dense chemical kinetic Jacobians using single-instruction, multiple-data (SIMD) and single-instruction, multiple thread (SIMT) paradigms. These are implemented in pyJac, an open-source, reproducible code generation platform. Selected chemical kinetic models covering the range of sizes typically used in reactive-flow simulations were used for demonstration. A new formulation of the chemical kinetic governing equations was derived and verified, resulting in Jacobian sparsities of 28.6–92.0% for the tested models. Speedups of 3.40–4.08 × were found for shallow-vectorized OpenCL source-rate evaluation compared with a parallel OpenMP code on an `avx2` central processing unit (CPU), increasing to 6.63–9.44 × and 3.03–4.23 × for sparse and dense chemical kinetic Jacobian evaluation, respectively. Furthermore, the effect of data-ordering was investigated and a storage pattern specifically formulated for vectorized evaluation was proposed; as well, the effect of the constant pressure/volume assumptions and varying vector widths were studied on source-term evaluation performance. Speedups reached up to 17.60 × and 45.13 × for dense and sparse evaluation on the GPU, and up to 55.11 × and 245.63 × on the CPU over a first-order finite-difference Jacobian approach. Further, dense Jacobian evaluation was up to 19.56 × and 2.84 × times faster than a previous version of `pyJac` on a CPU and GPU, respectively. Finally, future directions for vectorized chemical kinetic evaluation and sparse linear-algebra techniques were discussed.

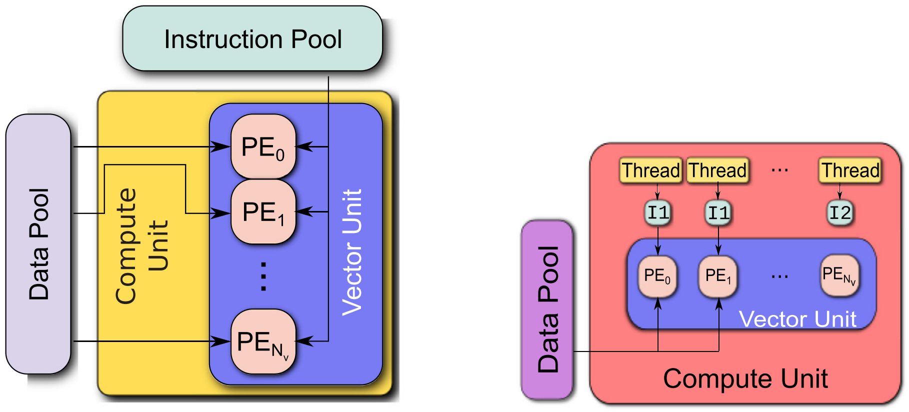
1. Introduction

As the combustion and reactive-flows community has recognized the importance of detailed chemical kinetics for predictive reactive-flow simulations [1], chemical kinetic models have grown in size and complexity to describe current and next-generation fuels relevant to transportation and power generation. For example, a recent biodiesel model [2] consists of 3500 chemical species and over 17000 reactions. Moreover, the cost of evaluating the chemical source-terms scales linearly with the size of the model, while evaluating and factorizing the chemical kinetic Jacobian respectively scale quadratically and cubically with the number of species in the model, if naively implemented via a finite-difference method [1]. These factors often prohibit using detailed chemical kinetics in practice; e.g., in a direct numerical simulation using a 22-species model, evaluating reaction rates consumed around half of the total run time [3]. In addition, most common implicit integration techniques need to evaluate and factorize the Jacobian matrix to deal with stiffness. As a result, these operations are bottlenecks when using even moderately sized chemical models in realistic reactive-flow simulations, necessitating other cost-reduction strategies [1].

A host of techniques have been developed to lessen the computational demand of chemical kinetic calculations while maintaining fidelity, falling broadly into three categories: removal of unimportant species and reactions [4–8], lumping of species with similar thermochemical properties [9–11], and time-scale methods that reduce numerical stiffness [12–15]. We refer interested

* Corresponding author.
E-mail address: nicholas.curtis@uconn.edu (N.J. Curtis).

https://doi.org/10.1016/j.combustflame.2018.09.008
0010-2180/© 2018 The Combustion Institute. Published by Elsevier Inc. All rights reserved.



(a) Schematic of SIMD processing. A single compute unit (e.g., a CPU core) contains a vector unit with N_v processing elements (PEs), together called a vector-lane. The vector unit executes a single instruction concurrently on multiple data.

(b) Schematic of SIMT processing. A single compute unit (e.g., a GPU streaming multiprocessor) contains many processing elements (PEs) and hosts many threads, each with an instruction to execute (I1, I2). Threads with the same instruction execute concurrently on multiple data while the others must wait (leading to thread divergence).

Fig. 1. Simple diagrams explaining the fundamentals of the SIMD and SIMT vector-processing paradigms.

readers to the recent review by Turányi and Tomlin [16] for a comprehensive overview.

In addition to the previously mentioned cost reduction methods, effort has gone into improving the integration algorithms and codes that evaluate the chemical kinetics [15,17–19]. In particular, a carefully derived analytical formulation of the Jacobian matrix can greatly increase sparsity [17] and drop the cost of Jacobian evaluation to linearly depend on the number of species in the model [1]; sparse-matrix techniques can then reduce the cost of Jacobian factorization [20]. In addition, studies have shown that Single-Instruction, Multiple-Data (SIMD) and the related Single-Instruction, Multiple-Thread (SIMT) processors can accelerate chemical kinetic simulations [18,21–25].

SIMD and SIMT programming are two important vector-processing paradigms used increasingly in scientific computing. Traditional multicore parallelism is now used to increase central processing unit (CPU) performance, as the exponential growth in processing power—colloquially known as Moore's law—has slowed [26]. Recently, SIMD/SIMT processors, e.g., in the form of graphics processing units (GPUs), have gained recognition due to their increased floating operation throughput. The parallel programming standard OpenCL [27] has further enabled adoption of vector processing in scientific computing by providing a common application program interface (API) for execution on heterogeneous systems, e.g., CPU, GPU, or Intel's Many Integrated Core (MIC) architecture. Here we will largely use OpenCL terminology to describe these processing paradigms, as it provides a convenient way to classify otherwise disparate processor types (e.g., CPUs and GPUs). However, the concepts discussed herein broadly apply to SIMD/SIMT processing.

A typical modern CPU contains multiple compute units (i.e., cores), each with specialized vector processing units capable of running SIMD instructions, as Fig. 1a depicts. A SIMD instruction uses the vector processor to execute the same floating-point operation (e.g., multiplication, division) on different data concurrently. The vector-width is the number of possible concurrent operations, typically around two to four in double precision.[1] Specialized hardware accelerators have also been developed, like Intel's Xeon Phi co-processor (i.e., the MIC architecture), that have tens of cores with wide vector-widths (e.g., 4–8 double-precision operations). Cutting-edge and forthcoming Intel CPUs also include these wide vector-widths, like the Skylake Xeon and Cannon Lake architectures.

Modern GPUs rely on the related computing paradigm of SIMT processing, where a single compute element hosts large numbers of threads (a streaming multiprocessor in Nvidia terminology) [28]. Figure 1b depicts a SIMT compute unit, where a group of threads—typically 32, known as a warp on Nvidia GPUs—execute the same SIMT instruction on multiple data concurrently. If some threads must execute a different instruction, they are forced to wait and execute later; this may occur due to if/then branching or predication. This phenomenon, known as thread-divergence, is a key consideration for SIMT processing and can cause serious performance degradation for complicated algorithms [24].

1.1. Related work

Recognizing the need to accelerate chemical-kinetic Jacobian evaluation and factorization, a number of recent works have been published on constructing analytical Jacobian matrices; although as will be discussed at the end of this section, here we offer several key improvements over past efforts. Schwer et al. [17] were among the first to recognize the critical importance of a sparse analytical Jacobian to accelerate chemical kinetic simulations. Later, Safta et al. [29] developed the `TChem` software package, which was one of the first developed that provides analytical Jacobian evaluation.

[1] OpenCL allows for use of vector-widths different from the actual hardware vector-width via implicit conversion, and may provide some performance benefit as Section 3.5 discusses.