超燃冲压发动机一维模型的GPU并行加速研究 *

温思歆[1]，苏承毅[2]，王东杰[1]，孟万植[1]，聂聆聪[2]，孙希明[1]

（1. 大连理工大学 控制科学与工程学院，辽宁 大连 116000；
2. 北京动力机械研究所，北京 100074）

摘 要：发动机模型是控制计划优化、基于模型的控制和观测器设计等技术的基础，对控制系统的性能具有重要影响。然而，超燃冲压发动机一维模型由于依赖计算流体力学的网格计算，其计算量巨大，难以在机载控制器内实时运行。为解决这一问题，本文深入研究基于图形处理器（Graphics Processing Unit，GPU）的并行计算技术，探索了网格解耦与划分、串/并异构设计、内存优化、代码优化、编译指令优化、硬件模式优化等方法，综合设计了一个高效的中央处理器（Central Processing Unit，CPU）+GPU异构模型，并在基于虚拟路径交叉连接（Virtual Path Cross-Connect，VPX）总线的嵌入式控制器上进行验证。为充分验证所设计异构模型的有效性、高效性与实时性，本文依次开展基线测试、软硬件优化加速测试、并行计算测试，在测试中对比一维模型在CPU，单核GPU，多核GPU上的计算耗时与数据误差。最后根据数据、曲线、监控工具等方式，展示了所设计的异构模型在不损失模型精度的前提下，加速超过了6.7倍，运行时间均不超过25 ms，符合工程预期的实时性要求，具有良好的应用前景。

关键词：超燃冲压发动机；并行计算；一维模型；嵌入式控制器；优化加速

中图分类号：V231.3 **文献标识码**：A **文章编号**：1001-4055（2024）10-2311031-10

DOI：10.13675/j.cnki. tjjs. 2311031

1 引 言

超燃冲压发动机作为高超声速飞行器的核心动力，在比冲、耗油率、推力等关键指标方面具有显著优势[1]，受到美国、俄罗斯、法国、德国、中国等军事强国的重视[2-3]。目前，发动机模型常被用于设计分布参数的观测器、供油系统的实时优化、故障诊断与容错控制等，在控制系统的设计过程中起到至关重要的作用[4]。但超燃冲压发动机是一个典型的强非线性、高动态、复杂流动的分布参数系统，其主流的建模方法是采用计算流体力学（Computational Fluid Dynamics，CFD）的网格计算[5]，具有运算复杂度高、耗时长等缺点，导致它难以在机载嵌入式控制器内进行实时计算，严重限制了基于模型的理论在超燃冲压发动机的工程应用[6]。因此，本文旨在提升超燃冲压发动机模型的计算效率，并在高性能控制器上验证，为基于模型的理论奠定基础。

近年来，国内外学者已经开展了大量超燃冲压发动机建模方法的研究[7-10]。比如，Ladeiende[8]回顾了超燃冲压发动机的模拟计算，并表明RANS方法占据湍流建模的主导地位。Choubey等[9]表明正确的、足够分辨率的网格是发动机模拟的主要关注点。Ma等[10]以实时仿真与控制为导向，提出了多种针对一维模型的简化策略。在众多研究中，通过二维、三维CFD数值模拟的发动机模型虽然有较高精度，但其计算规模大、耗时长，难以作为机载模型。此外，不同于传统涡轮发动机可以通过点代替部件进行建模，超燃冲压发动机是分布参数系统，实际试验通过传感器测量线状的沿程参数，因此其模型至少采用线状描述，难以简化为零维模型[11]。因此，在权衡计算精度与计算时间后，本文主要研究基于RANS方法的

* **收稿日期**：2023-11-12；**修订日期**：2024-01-16。
基金项目：国家科技重大专项（J2019-I-0019-0018）。
作者简介：温思歆，博士后，助理研究员，研究领域为发动机控制与优化。
通讯作者：孙希明，博士，教授，研究领域为航空发动机控制与故障诊断。E-mail：sunxm@dlut.edu.cn
引用格式：温思歆，苏承毅，王东杰，等. 超燃冲压发动机一维模型的GPU并行加速研究[J]. 推进技术，2024，45(10)：2311031.（WEN S X，SU C Y，WANG D J，et al. One-dimensional modeling of scramjet based on GPU parallel acceleration[J]. Journal of Propulsion Technology，2024，45(10)：2311031.）

一维简化模型，且模型的网格为500以内。

一维模型保留了满足工程应用的建模精度，也大幅降低了计算量，但实时运行于机载控制器仍是一项挑战。目前，最常见的一维模型是串行计算，其耗时取决于CPU的性能，因此许多学者采用高性能的酷睿I7处理器来降低耗时，如文献[10,12]，但这种方式成本高、硬件性能的上限低，具有一定的局限性。杨柳等[6]针对这个问题，提出采用多核数字信号处理器(Digital Signal Processor，DSP)的并行计算来提高计算效率，并在网格数为70的一维模型上进行验证，加速比达4倍以上，使得单个控制周期的模型计算耗时在30 ms以内。但是多核DSP的并行加速效益有限，若想进一步提高加速比，需要考虑多片DSP的集群式架构，其成本、设计复杂度、功耗较高，且重量明显增加。更有前景的高性能计算方式是基于图形处理器(Graphics Processing Unit，GPU)的并行计算，因为单个GPU芯片上含有多个流处理器，适合CFD这类网格计算，并已经得到应用[13-17]。比如Emelayanov等[13]使用了Tesla K40显卡，加速了20~50倍；Dobrov等[14]使用了Tesla P100显卡，加速了4.5~91.2倍；Rao等[15]使用了Tesla K40C显卡，加速了10~85倍；Liao等[16]使用了Tesla V100显卡，加速了11.26倍；赖剑奇等[17]使用了4块GTX1070显卡，加速了143倍。

尽管如此，一维模型的并行计算加速仍存在诸多技术问题。第一，GPU更适合于大规模网格的并行计算，针对小规模网格的加速效益不显著[17]，比如文献[13-17]都是采用高性能的显卡对逾百万的网格进行并行化，但加速效果有限。第二，高性能显卡拥有卓越的计算性能，但需与主机、电源、散热器等配套使用，具有功耗高、重量重、尺寸大等缺点，不符合航天领域的嵌入式控制器要求质量轻、尺寸小的需求。第三，发动机模型本质是迭代求解微分/差分方程，其中有大量参数需要串行计算，即具有严格的时序关系[7]，如何解决这种时序关系，实现网格解耦是开展并行计算的关键问题。第四，利用GPU计算一维模型时，需考虑网格划分到GPU线程、CPU与GPU之间协作/同步、GPU内存管理等关键问题[13-18]，设计不合理时将出现并行程序比串行程序更耗时的情况。由于这些障碍，现有研究仍难以应用在试车试验中。综上所述，在低端GPU上部署以加速基于数十、数百个网格的一维模型，实现毫秒级别的实时计算，是一项具有挑战性的工作。

为突破上述障碍，推动机载实时模型的工程应用，本文系统性地提出了一种超燃冲压发动机一维模型的并行计算方法，做出以下贡献：第一，本文将基于串行计算的一维模型转换为一个高效的CPU+GPU异构模型。具体地，本文针对一维模型的CFD网格计算，研究了网格解耦与划分、CPU+GPU异构设计、内存优化等方法，将原模型改进为适合GPU并行计算的结构。第二，本文针对集成TX2i与虚拟路径交叉连接(Virtual Path Cross-Connect，VPX)总线的嵌入式控制器，从代码、编译指令、硬件性能模式等角度对模型进行优化，进一步大幅缩短计算时间。第三，研究方法在VPX嵌入式控制器上进行验证。

2 方 法

2.1 基于串行计算的一维模型

为兼顾模型的计算速度与精度，本文所研究超燃冲压发动机的一维模型是以发动机轴向、隔离段-燃烧室内、均匀分布的沿程测量点来表征各个截面的平均特性。为刻画这些参数，一维模型基于流体的连续性、动量、能量守恒方程，构建如下的准一维微分方程[11]

$$\frac{\partial \boldsymbol{U}}{\partial t}+\frac{\partial \boldsymbol{F}(\boldsymbol{U})}{\partial x}=\boldsymbol{S}(\boldsymbol{U}) \tag{1}$$

式中x为沿程的相对位置，t为相对时间，$\boldsymbol{U}=\rho\left[A,Au,AE_{\mathrm{t}}\right]^{\mathrm{T}}$为守恒型矢量，$\boldsymbol{F}(\boldsymbol{U})=[\rho Au,p_{\mathrm{t}}A+\rho Au^2,(p_{\mathrm{t}}+\rho E_{\mathrm{t}})Au]^{\mathrm{T}}$为无粘通量项；$\boldsymbol{S}(\boldsymbol{U})=[\frac{\mathrm{d}\dot{m}_{\mathrm{f}}}{\mathrm{d}x},p_{\mathrm{t}}\frac{\mathrm{d}A}{\mathrm{d}x}-\frac{\rho u^2}{2}C_{\mathrm{f}}\cdot C_{\mathrm{w}}+u_x\frac{\mathrm{d}\dot{m}_{\mathrm{f}}}{\mathrm{d}x},(H_{\mathrm{v}}+H_{\mathrm{a}})\cdot\eta\cdot\frac{\mathrm{d}\dot{m}_{\mathrm{f}}}{\mathrm{d}x}-H_{\mathrm{l}}]^{\mathrm{T}}$为源项；$\rho$为流体密度，$A$为燃烧室流道横截面积，$E_{\mathrm{t}}$为流体总内能，$p_{\mathrm{t}}$为测量的流体总压，$m_{\mathrm{f}}$为燃料质量流量，$u$为气体流速，$u_x$为燃料喷注在$x$方向的流速，$C_{\mathrm{f}}$为壁面的摩擦力系数，$C_{\mathrm{w}}$为非圆形截面的湿周长，$H_{\mathrm{v}}$为标准条件下燃料的燃烧热值，$H_{\mathrm{a}}$为额外的热值增量，$H_{\mathrm{l}}$为壁面散热，$\eta$为燃烧效率分布。

在求解方程(1)时，采用兼具计算精度与计算效率的迎风格式CFD计算方法[19]。为缩短求解时间，模型的通量项用AUSM格式求解[17]，无粘通量分裂为对流项与压力项，源项采用一阶TVD Runge-Kutta格式进行求解，具体公式可参照文献[6,11]，本文不再赘述。

综上所述，一维模型的计算流程如图1所示。可见，该模型本质上是沿时间与空间的双重串行计算，它将流动区域沿轴向、等间距地划分成x_N个点。每个点x_k作为一个CFD网格的子模型。相邻网格有空

间前后的关联，即第 x_k 个网格的计算结果作为第 x_{k+1} 个网格的初始状态。在将所有网格迭代计算 t_{MAX} 次后，才得到一个控制周期的结果。假设发动机的采样周期为 T_s，则网格迭代的小步长时间 t_s 为

$$t_s = T_s / t_{\mathrm{MAX}} \tag{2}$$

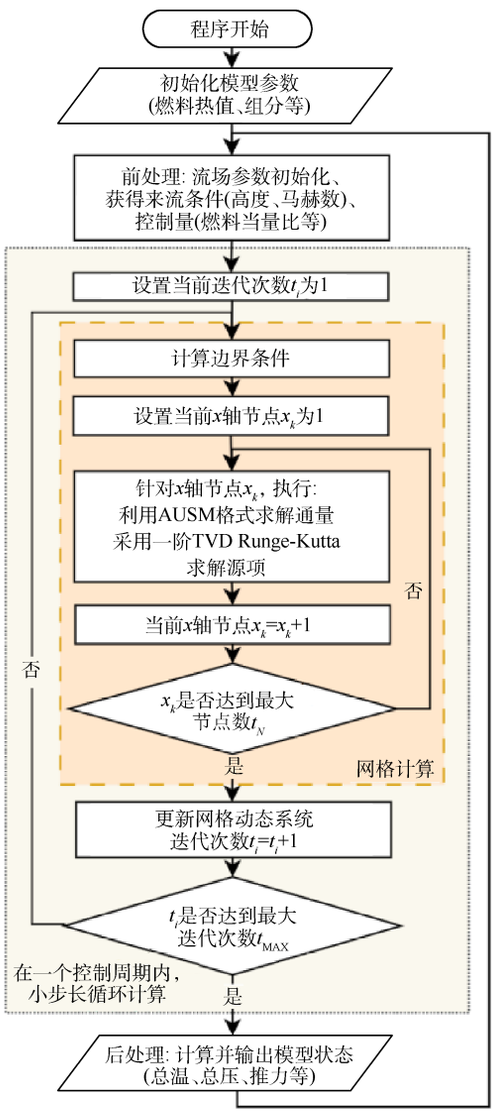

Fig. 1　Serial computing schematic for one-dimensional model of scramjet

为保证网格内模型的精度与计算稳定性，总迭代次数 t_{MAX} 与网格总数 x_N 成比例关系，即

$$t_{\mathrm{MAX}} = \lambda \cdot x_N \tag{3}$$

式中 x_N 是预先根据期望的空间分辨率设置的，如文献[6]设置为 70；λ 为比例系数，该系数与马赫数 Ma 相关。x_N 决定了 CFD 模型的空间分辨率，x_N 越大，空间分辨率越高，此时应迭代更多次来保证各个网格内子模型的精度。Ma 表征来流的流速，Ma 越大则说明来流的流速越快，相应地，t_s 应越小，此时 λ 应越大。Ma 决定了发动机隔离段-燃烧室的工况条件，如总温、总压等，本文采用的工况参数如表 1 所示。

Table 1　Model input conditions

Ma	Ma of isolator inlet	Total temperature/K	Total pressure/MPa	Fuel ratio of front injector/%
4	1.76	909	0.93	36
5	2.15	1 305	1.48	48
6	2.61	1 816	2.03	65
7	2.78	2 435	2.23	100

因此，在 x_N，Ma 确定的前提下，一维模型的总迭代次数是可预先确定的，即计算量是可估计的。假设一个网格计算一次的复杂度为 $O(n)$，则一维发动机模型在一个控制周期内的计算量达 $O\left(nt_{\mathrm{MAX}}x_N\right) = O\left(n\lambda x_N^2\right)$。因此，当空间内选取节点较多时，需要耗费大量计算时间。

2.2　嵌入式控制器

针对超燃冲压发动机机载模型求解过程计算量大的问题，北京动力机械研究所研制了基于 VPX 架构的高性能嵌入式控制器。该控制器如图 2 所示，包含了电源模块、参数采集及控制模块、通信模块、GPU 计算模块。其中，GPU 计算模块是以英伟达的 TX2i 为核心，内含 2 核的 Denver2 型号 CPU，4 核的 Arm Cortex-A57 型号 CPU，每个 CPU 核的主频为 2 GHz；还包含 256 个 CUDA 核，每个 CUDA 核的频率为 1.3 GHz；单精度峰值性能高达 1.26 TFLOPS[19]。相比之下，多核的 FT-M6678 型号 DSP 仅拥有 8 个核心，单核主频为 1 GHz。可见，集成 TX2i 的 VPX 控制器拥有优异的计算性能，适合运行发动机模型这类计算密集型的任务。因此，本文基于该 VPX 控制器进行一维模型的并行计算验证。

在软件层面，VPX 控制器运行了开源的 Ubuntu

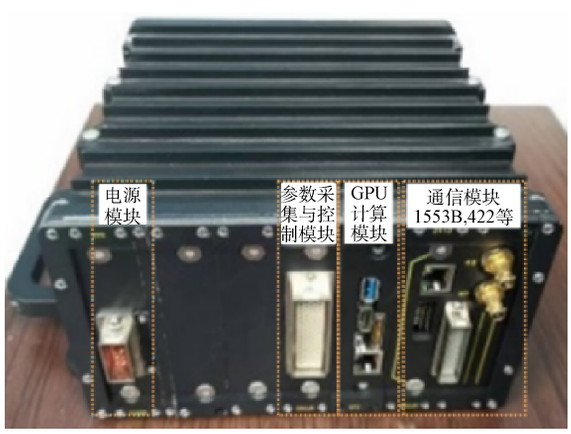

Fig. 2　VPX embedded controller

系统，并利用英伟达公司提供的CUDA应用编程接口来调用GPU资源，使用C/C++编程语言的语法即可开发GPU与CPU程序[17]。CUDA将GPU作为CPU的协处理器进行异构计算，从而高效率地调动VPX的计算资源[20]。

2.3 并行方法

超燃冲压发动机的一维模型需要在一个控制周期内进行时间、空间的迭代计算，耗费时间过长。本节重点研究针对该一维模型的并行计算方法，以缩短计算时间。

2.3.1 网格解耦与划分

由图1可知，一维模型将发动机分为x_N个网格进行计算，因此本文并行计算的基本思想是将网格分配到GPU线程，然后调用核函数进行单指令多数据的计算。然而，串行模型的相邻网格存在空间关联性，即在每个迭代周期t_i内，第x_{k+1}个网格依赖第x_k个网格的计算结果，采用如式(4)表示此过程，即

$$g\left(x_{k+1},t_i\right)=f\left(g\left(x_k,t_i\right)\right) \tag{4}$$

式中f表示网格内模型的计算函数，g表示网格内模型的输出，包含静温、静压、流速、源项等参数。这种空间关联性要求各个网格必须顺序地执行，导致模型难以进行并行计算。为消除该关联性，本文令第t_i次迭代的x_{k+1}网格的初始状态为第t_{i-1}次迭代的x_k网格的计算结果，采用如式(5)来直观地描述该思路，即

$$g\left(x_{k+1},t_i\right)=f\left(g\left(x_k,t_{i-1}\right)\right) \tag{5}$$

即每个网格的输入为上一个时刻、前一个网格的计算结果。采用这样的方式，在第t_i次迭代的各个网格是互相独立的，因此可进行并行计算。值得注意的是，该方法将导致并行模型比串行模型在时间轴t上慢了一个迭代周期，使得两个模型的计算结果存在一定差异，这种差异是难以避免的。但鉴于每次迭代的时间步长较短，发动机模型在如此短时间内的变化较小，由此产生的小偏差可被接受，后文将以测试结果证实该偏差的具体数值。

将网格分配给不同线程主要有两种划分方式，即分为网格块、独立网格。划分为独立网格，则每个网格拥有一个独立的线程，线程间是相对独立的，但可通过全局内存、共享内存等方式共享线程数据。这种方式并行度高，适合于网格数少、网格间共享数据量小的情况。本文的研究是针对嵌入式机载的实时应用，在综合考虑计算精度与实际工程需求后，要求网格总数仅为500，且网格间仅需共享公式(5)所代表的g参数，共享数据量小。同时，考虑到线程数越多越能发挥GPU性能，本文将一维模型的每个网格分配给单独的线程，由GPU调用CUDA核进行并行计算。

2.3.2 异构计算架构设计

发动机一维模型不仅包含可并行的网格计算，还包含许多需要串行计算的部分。如图1所示，一维模型在每个控制周期内需执行前处理与后处理[11]，其中前处理主要完成流场参数初始化、参数单位转换等，后处理是对网格参数进行汇总处理，以得到推力、摩擦力、总温、总压、稳定裕度、比冲等。由于公式较多与限于篇幅，具体公式可查看文献[11]。此外，每一个网格进行小步长地迭代计算时，需要利用上一次t_{i-1}迭代的结果来计算当前t_i迭代的边界条件，并且在更新网格动态系数时需汇总所有网格的计算结果，导致从时间轴t上看，网格是串行计算的。综上所述，在一维模型的计算流程中，除了在第t_i次迭代的多个网格可进行并行计算外，其他部分仍需串行计算。因此，本文需将一维模型内的计算任务合理地分配到CPU，GPU，形成高效的异构计算架构。

在VPX控制器的串行计算方面，相比于单核GPU，单核CPU具有较低的延迟、较快的内存访问速度、更优的编译器指令，即CPU具有较高的单线程计算性能，后文将给出二者的测试结果来证实此结论。此外，不同于CPU在调用函数时仅需执行压栈、出栈等简单操作，GPU在启动内核函数时有较大的开销，因为它需传输数据、创建网格、线程块与线程，线程还需被GPU调度、分配给SM，然后才能执行程序[18]。当内核函数的计算规模较小时，这种额外开销将对加速性能产生显著的负面影响。因此，鉴于(1)在串行计算方面，CPU性能明显优于单核GPU；(2)所研究一维模型的网格数最大仅为500，导致前处理与后处理的计算规模较小；本文在CPU里执行一维模型的前处理、后处理计算。

如图1所示，在计算x_N个网格的前、后需分别计算边界条件、网格动态系数[11]。若将这些程序分配给CPU执行，则每次调用GPU内核函数时仅执行一次网格计算，导致在一个控制周期内需调用t_{MAX}次内核函数，这将产生大量内核函数的启动开销。为避免这种情况，本文将网格计算的整个迭代过程都部署在内核函数里，通过加入线程同步函数来严格保证时序的正确性，即在边界条件之后、在网格动态系数之前都进行同步。如此，每一个控制周期仅需启动一次内核函数，减少了不必要的内核函数启动开销，并且无需在CPU与GPU之间频繁传输数据，减少

了数据传输的开销与延迟。

2.3.3　内存优化

异构计算架构要求在执行内核函数前需将 CPU 数据传输至 GPU，执行后将 GPU 数据拷贝至 CPU。本文所研究的一维模型包含静温、静压、总温、总压、流速、源项、通量等多个参数，若将每个参数分别传输，则需多次调用内存拷贝函数，增加了函数调用开销与延迟[21]。为避免该问题，本文利用结构体打包数据，然后一次性传输。如此，在一个控制周期内仅需调用两次内存拷贝函数，有效减少数据传输的开销和延迟[15]。此外，通过结构体的组织形式可方便读写数据，并具有良好的可读性。

充分利用 GPU 的内存结构可减少数据读写的开销。GPU 与 CPU 之间传输的模型参数都被存储至全局内存，VPX 控制器的全局内存空间达 8 GB，但读写速度最慢。此外，在访问结构体内的变量时，需根据结构体起始地址及该变量的偏移量来计算物理地址，然后根据该地址来访问变量数值，这个过程将产生一定开销。因此，本文在 GPU 端为结构体内的每个变量都分别创建了新的变量，通过直接访问变量的形式来提高内存读写效率。进一步地，比全局内存更快的是寄存器、局部内存、共享内存。但寄存器在所有线程中仅有 256 kB，均分到 500 个网格的线程中，每个线程仅有 524 Bytes；共享内存在每个 Block 最大可用 48 kB。因此，本文优先将读写最频繁的通量、源项、静温、静压、流速等参数分配到寄存器，将需要用于计算边界条件、网格动态系数的数据分配到共享内存，其他数据则存储于局部内存。其中寄存器、局部内存是线程内的私有变量，则每一个网格内的静温、静压等私有参数只需要分配为变量，以减少内存空间的占用。

综上所述，超燃冲压发动机一维模型的异构计算架构如图 3 所示，它通过综合上述的网格解耦与划分、CPU+GPU 异构设计、内存优化等方法，在最大程度上提高模型的计算效率，其有效性将通过下一节的多个例子进行说明。

3　结果与讨论

3.1　基线测试

本文所研究的串行模型是通过 C 语言编程实现的，并已结合试验数据验证了该模型具有良好的建模精度[6]，但该串行模型计算速度较慢。该 C 语言程序可方便地移植到 VPX 控制器的 Ubuntu 系统里，并可将运行时间、静温、静压、推力等结果保存至 CSV 文件，方便进行统一的数据处理与分析。其中运行时间是记录从前处理运行之前到后处理结束时的时间，即单个控制周期内模型计算的总耗时。此外，文中所展示的时间均为 100 个运行周期的平均值，单位

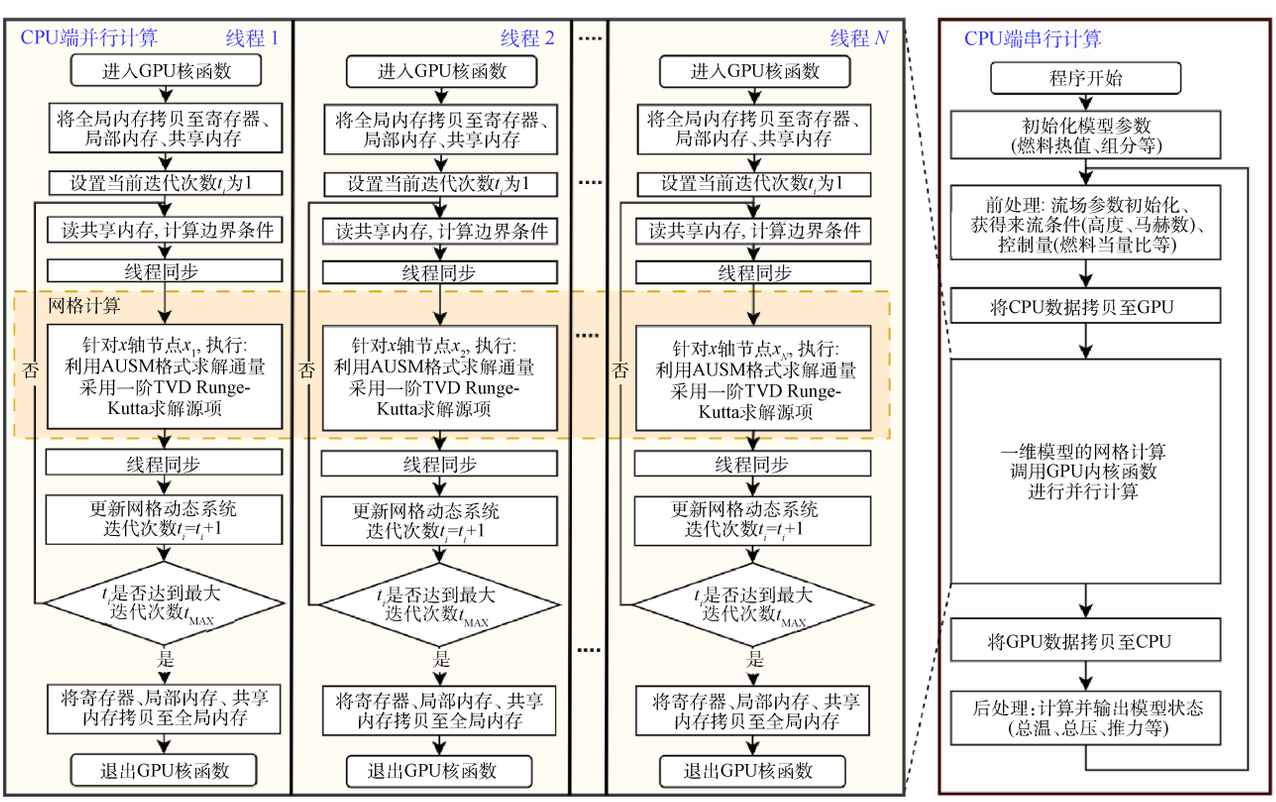

Fig. 3　Parallel computing schematic for one-dimensional model of scramjet

均为ms。

GPU的单核主频、指令集、内存与缓存结构等方面与CPU有较大差距，并且不同级别、架构、成本的CPU与GPU存在较大的性能差异，因此难以公平地对比加速性能。为此，本小节给出单核GPU（Single-core GPU）运行串行模型的结果，将该结果作为基准，以便在统一的硬件架构上，更公平地说明所研究方法的有效性与加速效益。从而突出多核并行计算的优势，也更直观地表明多核GPU的计算瓶颈。

由于一维模型运行在CPU，单核GPU的区别仅为程序运行在不同硬件平台，计算的精度、数值等结果不会受到影响。因此，本小节仅给出耗时的对比结果，该结果如表2所示，可以得出以下结论：（1）在x_N为500时，CPU的耗时远远超过了期望的25 ms，即不满足实时性要求；（2）模型耗时与x_N，Ma呈现正相关的关系，符合第2.1节的建模原理；（3）当x_N从70增加至500时，串行模型的耗时比接近51倍，与前文分析的计算复杂度$O\left(n\lambda x_N^2\right)$相匹配；（4）CPU的计算速度比单核GPU快了超过10倍，表明串行计算的程序更适合由CPU执行，表明第2.3.2小节设计异构计算架构的重要性；（5）在单核GPU性能较低的前提下，研究多核GPU并行计算以大幅缩短计算时间，在耗时上少于串行模型，具有一定的挑战性。

3.2　软、硬件优化加速结果

前一小节的基线模型未执行任何优化，导致耗时过长。为缩短计算时间，本小节从以下三个角度对一维模型的程序进行优化。

第一，代码优化：将频繁调用且代码量较少的源项求解、线性转换等函数修改为宏定义形式，将频繁调用且代码量较多的通量求解、燃烧热值计算等函数声明为内联函数，从而减少函数调用的开销；将计算的中间值保存下来以避免重复计算，提高代码执行效率；给所有浮点常量后面加上“f”，实现显式类型声明，避免不必要的隐式类型转换的计算开销。

第二，编译指令优化：将编译器的优化选项从默认的“-O0”修改为“-O3”，利用高优化等级来提升程序运行速度；由于模型中包含大量的开平方、求幂次方等操作，启用快速数学库“-use_fast_math”选项；选定指定目标GPU架构为“sm_62”，提高代码的性能与兼容性。

第三，硬件模式优化：通过命令“sudo nvpmodel -m 0”，“sudo jetson_clocks”指令强制启用CPU，GPU的最大性能模式，因为性能模式默认会平衡性能与功耗，导致无法最大化发挥硬件的性能。但实际上，该过程产生的功耗、热量与机载机电系统相比是微不足道的。

以上三步的方法对模型数据的影响较小，所以本节仅给出优化后的耗时结果。结果如表3所示，与基线测试的表2进行对比，可以看出：（1）优化后的模型耗时显著降低，其中CPU的结果加速超过6.4倍，单核GPU的结果加速超过3.1倍；（2）在x_N从70增加至500时，优化后模型耗时增加了超过46.6倍，说明所采用的优化手段难以抑制由x_N增加引起的巨大耗时；（3）从串行计算的角度优化单核模型的计算已经达到瓶颈，但x_N为500的结果仍然无法满足25 ms的实时性要求，因此有必要从并行计算角度进行优化；

Table 2　Timing results of the baseline cases　　**(ms)**

Ma	CPU			Single-core GPU		
	x_N=70	x_N=500	Time ratio	x_N=70	x_N=500	Time ratio
4	12.51	631.01	**50.4x**	161.88	7 826.43	**48.3x**
5	16.59	836.45	**50.4x**	205.85	10 038.15	**48.8x**
6	22.07	1 099.18	**49.8x**	259.26	12 703.65	**49.0x**
7	27.78	1 413.77	**50.9x**	313.72	15 403.25	**49.1x**

Table 3　Timing results of the optimized cases　　**(ms)**

Ma	CPU				Single-core GPU			
	x_N=70		x_N=500		x_N=70		x_N=500	
	Time	Acc ratio	Time	Acc ratio	Time	Acc ratio	Time	Acc ratio
4	1.93	**6.5x**	91.40	**6.9x**	51.04	**3.2x**	2 435.19	**3.2x**
5	2.50	**6.6x**	114.34	**7.3x**	63.90	**3.2x**	3 080.39	**3.3x**
6	2.97	**7.4x**	145.01	**7.6x**	78.72	**3.3x**	3 805.11	**3.3x**
7	3.34	**8.3x**	162.40	**8.7x**	92.38	**3.4x**	4 492.93	**3.4x**

(4)经过优化后,CPU的计算速度比单核GPU快了超过26.2倍,显著增加了多核GPU并行计算超越CPU串行计算的难度。

3.3　多核并行计算结果

在对模型进行并行化之前,需要分析出模型中最耗时的部分。本文以x_N为500,CPU串行的模型为例子(记为Case 1),仅记录网格计算部分的时间,结果如表4所示。该结果与表3相比,可发现网格计算所耗时间占模型总耗时的98.4%以上,是模型中最耗时的部分。因此,本文主要针对模型的网格计算进行并行化。

值得注意的是,仅根据第2.3.1,2.3.2小节的方法在GPU线程里计算各网格,没有优化内存,得到结果仍不能满足25 ms的实时性要求,此结论以x_N为500的多核GPU并行计算为例子(记为Case 2)进行说明,记录整个模型的计算时间,结果在表4里展示。可以看出:(1)并行后模型的耗时显著缩短了,比单核GPU的结果快了超过60.2倍,比CPU的结果快了超过2.2倍;(2)但并行结果仍然超过预期的25 ms,不能满足预期25 ms的工程需求。

为进一步减少模型的耗时,本文使用第2.3节中全部的并行方法将原模型转换为一个高效率的异构模型。本小节以x_N为500的多核GPU并行计算为例子(记为Case 3)进行说明,时间统计结果在表4里展示。从表4与表3的对比结果可以看出:(1)最终模型的耗时均在25 ms以内,满足预期要求;(2)在加速性能方面,最终模型比单核GPU的结果快了超过187.5倍,比CPU的结果快了超过6.7倍。

通过GPU并行的耗时结果可以说明,本文方法有效利用了GPU资源,但无法表明第2.3节方法的有效性。为此,本文利用nvprof工具来对结果进行细致的分析,该工具为CUDA程序提供了详细的统计信息[21]。以Case 3运行1 000个控制周期为例进行说明,nvprof的监测结果如表5所示。可看出:(1)内核函数仅被调用1 000次,与第2.3.2小节所述的将整个迭代过程都部署在内核函数的方法相匹配;(2)GPU内存申请函数(cudaMalloc)仅被调用1次,从GPU至CPU、从CPU至GPU的内存拷贝函数仅被分别调用1 000次,与第2.3.3小节所述的将数据打包的方法相匹配;(3)内核函数的运行时间是有波动的,最大波动量在1.06 ms以内,但在工程中经常留有一定的空闲时间,该波动量在接受范围内。

由于并行模型在网格解耦时存在一个迭代周期的错位,导致结果与并行模型有微小的偏差。本文以x_N为500,Ma为7为例进行说明,图4展示静压、静温在空间的分布,图5展示静压、静温、推力在时间的响应。可以看出,两个模型在时间、空间的差异较小,曲线几乎重合。对测试数据进行统计分析,静压的最大误差为6.2×10^{-5},静温的最大误差为2.9×10^{-5},推力的最大误差为7.5×10^{-6},这些误差甚至小于传感

Table 4　Timing results of three cases　　**(ms)**

Ma	Case 1 (Only grid time)	Case 2		Case 3	
		Time	Acc ratio	Time	Acc ratio
4	89.98 (98.4%)	40.44	**2.3x**	12.94	**7.1x**
5	112.59 (98.4%)	50.87	**2.2x**	16.35	**7.0x**
6	142.88 (98.5%)	62.55	**2.3x**	20.28	**7.1x**
7	159.94 (98.5%)	73.66	**2.2x**	23.96	**6.8x**

Table 5　Monitoring results using the nvprof tool

Time/%	Time	Calls	Avg.	Min.	Max.	Function name
99.96	23.619 s	1 000	23.619 ms	23.534 ms	24.593 ms	SolveKernal (MEM_BLOCK*)
0.03	6.177 ms	1 000	6.177 μs	5.760 μs	7.200 μs	[CUDA memcpy HtoD]
0.01	2.507 ms	1 000	2.506 μs	2.304 μs	3.584 μs	[CUDA memcpy DtoH]
97.76	23.615 s	1 000	23.615 ms	23.526 ms	24.582 ms	cudaEventSynchronize
1.04	251.060 ms	2	125.530 ms	209.250 μs	250.850 ms	cudaFree
0.82	197.480 ms	2 000	98.738 μs	49.792 μs	2.079 ms	cudaMemcpy
0.23	56.102 ms	1 000	56.102 μs	34.688 μs	113.630 μs	cudaLaunchKernel
0.12	28.428 ms	2 000	14.214 μs	11.136 μs	43.680 μs	cudaEventRecord
0.03	6.545 ms	1 000	6.545 μs	3.136 μs	10.528 μs	cudaEventElapsedTime
0.00	639.680 μs	1	639.680 μs	639.680 μs	639.680 μs	cudaMalloc

器测量、量化噪声，可被接受。通过这个例子，证实了本文解耦网格的方法、CUDA 并行程序是正确的。

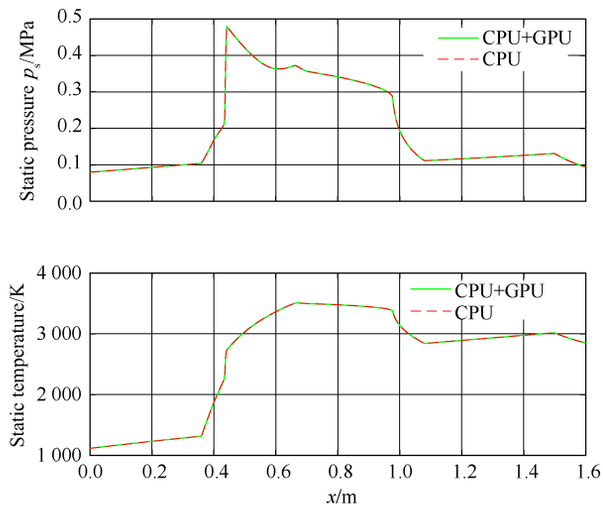

Fig. 4　Distribution of static pressure and static temperature along the engine axis

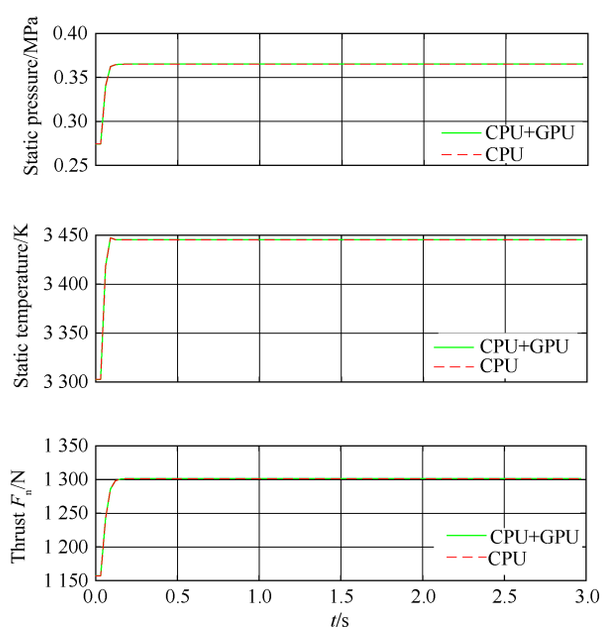

Fig. 5　Distribution of static pressure, static temperature and thrust along the time axis

根据第 2.1 节的分析，一维模型的计算耗时主要与 x_N，Ma 有关。因此，为全面地验证所设计异构模型的加速性能，本文令 x_N 依次为 70，140，280，500，令 Ma 依次为 4，5，6，7，进行多次测试。试验结果如表 6 所示，可以看出：(1)并行模型的耗时均在 25 ms 以内，满足实时性要求；(2)并行模型的耗时结果全面超越了 CPU 模型，并且网格数越多，加速效果越明显；(3)在 x_N 从 70 到 140，280，500 的过程中，并行模型耗时是 x_N 为 70 的约 2.4，7.0，18.6 倍，打破了原来 x_N^2 的耗时关系，降低了计算复杂度。因此，测试结果充分表明并行计算的优越性与本文所研究方法的高效性。

Table 6　Timing results of the CPU+GPU heterogeneous model　(ms)

Ma	x_N=70	x_N=140	x_N=280	x_N=500
4	0.73	1.70	5.06	12.94
5	0.88	2.15	6.09	16.35
6	1.30	2.84	7.59	20.28
7	1.32	3.19	9.18	23.96

4　结　论

本文针对超燃冲压发动机一维模型的串行计算速度慢问题，开展基于 GPU 并行计算的研究，并在 VPX 嵌入式控制器上进行了多个对比测试，得到以下结论：

(1)本文从代码、编译指令、硬件模式等角度对 CPU 串行模型进行优化，通过与基线测试的对比，证实了所采用的优化方法能够显著提高计算效率。

(2)本文所提出的网格解耦方法能够在几乎不损失模型精度的同时，将串行计算的网格转换为可并行计算的结构。在此基础上，本文设计出了一个高效的异构模型，该模型在网格数为 500、来流马赫数为 7 的情况下，计算速度与优化过的 CPU，单核 GPU 模型相比，分别提升了 6.7 倍、187.5 倍，有效降低了原串行模型的计算复杂度，具有更优异的计算性能。

但本文的研究仅针对燃烧室串行模型，为应用到高超声速飞行器，还需开展进气道建模、激波修正、数据修模等研究。

致　谢：感谢国家科技重大专项的资助。感谢北京动力机械研究所张永亮博士提供 VPX 控制器、牟春晖博士提供的建模指导。

参考文献

[1] CHOUBEY G, DEVARAJAN Y, HUANG W, et al. Recent advances in cavity-based scramjet engine-a brief review [J]. International Journal of Hydrogen Energy, 2019, 44(26): 13895-13909.

[2] 纪鉴恒，蔡　尊，王泰宇，等. 宽速域超燃冲压发动机流动燃烧过程研究进展[J]. 航空学报，2023，44(8)：528696.

[3] 袁春飞，仇小杰. 超燃冲压发动机研究现状及控制系统关键技术[J]. 航空发动机，2016，42(4)：1-7.

[4] 孙明波，安　彬，汪洪波，等. 超燃冲压发动机仿真：从数值飞行到数智飞行[J]. 力学学报，2022，54(3)：588-600.

[5] 姚　卫，张　政，赵　伟，等. 高超声速飞/发一体化进展与趋势[J]. 推进技术，2023，44(8)：22010002.（YAO W，ZHANG Z，ZHAO W，et al. Advances and trends in airframe/engine interation of hypersonic vehicles[J]. Journal of Propulsion Technology，2023，44(8)：22010002.）

[6] 杨　柳，史新兴，刘小勇，等. 基于多核并行计算的超燃冲压发动机一维模型实时性研究[J]. 推进技术，2022，43(6)：353-360.（YANG L，SHI X X，LIU X Y，et al. Real-time research of scramjet one dimensional model simulation based on multi-core parallel computation[J]. Journal of Propulsion Technology，2022，43(6)：353-360.）

[7] 高　峰，王宏宇，张　涵. 超燃冲压发动机燃烧室流场数值分析研究综述[J]. 飞航导弹，2014(1)：80-84.

[8] LADEIENDE F. A critical review of scramjet combustion simulation[C]. Florida：47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition，2009.

[9] CHOUBEY G，YADAV P M，DEVARAJA Y，et al. Numerical investigation on mixing improvement mechanism of transverse injection based scramjet combustor[J]. Acta Astronautica，2021，188：426-437.

[10] MA J C，CHANG J T，ZHANG J L，et al. Control-oriented modeling and real-time simulation method for a dual-mode scramjet combustor[J]. Acta Astronautica，2018，153：82-94.

[11] 马继承. 碳氢燃料再生冷却超燃冲压发动机控制方法研究[D]. 哈尔滨：哈尔滨工业大学，2021.

[12] LI N，ZHAO L M，BAO C Y，et al. A real-time information integration framework for multidisciplinary coupling of complex aircrafts：an application of IIIE[J]. Journal of Industrial Information Integration，2021，22：100203.

[13] EMELAYANOV V，KARPENKO A，VOLKOV K. Simulation of hypersonic flows with equilibrium chemical reactions on graphics processor units[J]. Acta Astronautica，2019，163：259-271.

[14] DOBROV Y，KARPENKO A，MALKOVSKY S，et al. Simulation of high-temperature flowfield around hypersonic waverider using graphics processor units[J]. Acta Astronautica，2023，204：745-760.

[15] RAO S H，XU X，CHEN B，et al. An investigation of hybrid CPU-GPU solvers for supersonic reacting flow simulation with detailed chemical kinetics[J]. Aerospace Science and Technology，2022，126：107597.

[16] LIAO Z X，LIU Y Z，CHE Y G. GPU parallelization and optimization of a combustion simulation application[C]. Chengdu：2022 IEEE 24th Int. Conf. on High Performance Computing & Communications，2022：67-73.

[17] 赖剑奇，李　桦，张　冉，等. 多GPU并行可压缩流求解器及其性能分析[J]. 航空学报，2018，39(9)：26-35.

[18] COOK S. CUDA programming：a developer's guide to parallel computing with GPUs[M]. Massachusetts：Elsevier Newnes，2012.

[19] TORO E F. Riemann solvers and numerical methods for fluid dynamics：a practical introduction[M]. Third edition. New York：Springer，2009.

[20] ADAMS C，SPAIN A，PARKER J，et al. Towards an integrated GPU accelerated SoC as a flight computer for small satellites[C]. Montana：2019 IEEE Aerospace Conference，2019：1-7.

[21] CHENG J，GROSSMAN M，MCKERCHER T. Professional CUDA C programming[M]. State of New Jersey：John Wiley & Sons，2014.

（编辑：梅　瑛）

One-dimensional modeling of scramjet based on GPU parallel acceleration

WEN Sixin[1], SU Chengyi[2], WANG Dongjie[1], MENG Wanzhi[1], NIE Lingcong[2], SUN Ximing[1]

(1. School of Control Science and Engineering, Dalian University of Technology, Dalian 116000, China;
2. Beijing Power Machinery Institute, Beijing 100074, China)

Abstract: The engine model serves as the foundation for various technologies such as control plan optimization, model-based control, and observer design, all of which significantly impact the performance of control systems. However, the computational requirements of one-dimensional models for scramjets are immense, making real-time execution on onboard controllers challenging. To address this issue, this study delves into the research of GPU (Graphics Processing Unit)-based parallel computing techniques and explores methods such as grid decoupling and partitioning, serial/parallel heterogeneous design, memory optimization, code optimization, compilation instruction optimization, and hardware mode optimization. By integrating these approaches, an efficient CPU (Central Processing Unit)+GPU heterogeneous model is designed and validated on the embedded controller based on VPX (Virtual Path Cross-Connect) bus. To adequately verify the effectiveness, efficiency, and real-time performance of the designed heterogeneous model, baseline tests, hardware and software optimization acceleration tests, and parallel computing tests are conducted in this paper. In the tests, the time consumption and data errors of the one-dimensional model on CPU, single-core GPU, and multi-core GPU are compared. Finally, leveraging data analysis, graphical representations, and monitoring tools, the study conclusively demonstrates that the designed heterogeneous model achieves an acceleration exceeding 6.7 times without compromising model accuracy. Importantly, none of the execution times surpass 25 ms, aligning with the real-time requirements essential for engineering applications. The methodologies investigated in this study showcase promising prospects for practical implementation.

Key words: Scramjet; Parallel computing; One-dimensional model; Embedded controllers; Optimal acceleration

Received: 2023-11-12; **Revised**: 2024-01-16.

DOI: 10.13675/j.cnki. tjjs. 2311031

Foundation item: National Science and Technology Major Project of China (J2019-I-0019-0018).

Corresponding author: SUN Ximing, E-mail: sunxm@dlut.edu.cn