A programming framework for large scale numerical simulations on unstructured mesh

QingKai Liu, WeiBo Zhao, Jie Cheng, ZeYao Mo, AiQing Zhang, JianJun Liu
Institute of Applied Physics and Computational Mathematics
CAEP Software Center for High Performance Numerical Simulation
Beijing, China
zhao_weibo@iapcm.ac.cn, cheng_jie@iapcm.ac.cn

Abstract—The performance of parallel computers grows rapidly. However, application software is lagging behind due to two bottlenecks: "performance wall" and "programmability wall". These bottlenecks have prevented a lot of application software from achieving good performance and fast development. Programming framework is considered an effective approach to overcome the above bottlenecks. In this paper, we give a prototype of JAUMIN, which is a programming framework for large scale application software based on unstructured mesh. Some important technologies of JAUMIN will be presented, including distributed data structures, data communication patterns and application programming interfaces. Finally, some applications based on JAUMIN will be demonstrated to show that JAUMIN can accelerate the development of application software greatly and support effective simulation on petascale supercomputer.

Keywords—numerical simulation; parallel computing; JAUMIN; programming framework; unstructured mesh

I. INTRODUCTION

Supercomputer is the fundamental infrastructure for high performance computing. In the past decades, the computational power of supercomputer have grown rapidly. The peak floating point operation performance of the most advanced supercomputer achieves 5.49PFlop/s [1]. The rapid growth of computing capacity of supercomputers makes solving extreme scale real-world problems possible. These problems feature multi-scale and multi-physics coupling. For examples, the Inertial Confinement Fusion (ICF) simulations include physical processes such as neutron transport, hydrodynamics instability, multi-group diffusion, and so on [2]. These physical processes are intrinsically difficult to simulate. Development of the software for these problems is a huge challenge.

On the other hand, as the computational power grow rapidly, the architecture of supercomputers is getting more and more complicated. Tianhe-2, which ranked first in the TOP500 list for six consecutive times, has a hierarchical architecture. The whole computing system of Tianhe-2 is made up of 125 cabinets. Each cabinet contains four frames, and each frame has 16 computing board. On each computing board, there are two computing nodes, each of which comprise two Intel Ivy Bridge Xeon processors and three Xeon Phi coprocessor chips. Programming on these supercomputers is very difficult for a domain expert, who have to deep into the hardware architecture to achieve good performance.

The dual complexity of both supercomputers and applications poses great challenges for the development of parallel application software [3]. We have the well-known performance wall and programmability wall. The performance wall means the low computing efficiency. It includes floating point operation performance on one CPU and the parallel scalability on the parallel system. The programmability wall means the long development cycle. It may take several years to develop an application software for the real-world problems. In this process, the architecture of supercomputer may have undergone great changes. So the code is very difficult to run efficiently on the new computers. To conquer these walls, we need a new way to develop application software.

The framework approach is considered a promising paradigm for the development of extreme scale applications [4], [5], [6]. In fact, as shown in Fig. 1, almost every numerical simulation software is composed of 3 parts: The computer science part of HPC implementations, the mathematics part of numerical algorithms, and the physics part of application models [7]. The two parts below are domain common technologies, and can be reused in multiple applications. Based on appropriate software engineering technologies, the two parts below can be integrated into a framework. In the framework, object-oriented programming interfaces are provided for the physical models development. Based on the framework, application experts can build parallel programs, and don’t have to understand the details of the HPC implementation. So, the application development can be greatly simplified.

![Fig. 1. The architecture of high performance numerical simulation software.](image-url)
Currently, the programming framework approach is widely explored in the world. Frameworks based on structured mesh, such as SAMRAI [8], JASMIN [4], Uintah [9], ParaMesh [10], are used widely in computational fluid dynamics, molecular dynamics simulation, combustion simulation, and so on. Frameworks based on unstructured mesh, such as SIEERA [11], [12], UG [13], PUMI [14], are used widely in structural mechanics analysis, thermal mechanics analysis, design optimization, etc.

The contribution of this article is to present a prototype of an unstructured mesh framework called JAUMIN. This framework features compound functionalities such as halo-exchange based computing, digraph based data-driven sweeping, contact-impact simulation, as well as linear and eigen solver support. It provides with component based application user interfaces which hide the parallel details from application developers. The framework highlights itself with its ability to shorten development cycle of applications while allowing them to achieve high performance on petascale super computers.

The remaining sections are organized as follows. In section II, we give details into our JAUMIN framework, including background and motivation of its development, its technical details like mesh data structure, field data management, communication patterns, and the design of application interfaces. In section III, we present several application software developed on JAUMIN, with particular emphasis on fast development and good computational performance. In section IV, we make some conclusions of our framework.

II. JAUMIN FRAMEWORK

JAUMIN (J Parallel Adaptive Unstructured Mesh Applications Infrastructure) is a parallel programming framework for unstructured mesh applications. It targets at real-world applications such as structural analysis, impact dynamics, neutron transport, etc. It aims to hide the complexity of parallel programming from application experts and support quick development of extreme-scale applications on personal computers. JAUMIN achieves this goal by providing efficient data structures and highly optimized parallel algorithms, as well as carefully designed user interfaces.

JAUMIN has been developed since early 2011. We have released a primary version, on which tens of applications are being actively developed.

A. Architecture design

The architecture design of JAUMIN follows the layering paradigm. Fig. 2 shows the software architecture of JAUMIN. It can be divided into three layers. The bottom layer consists of modules for high performance computing for unstructured meshes. In this layer, modules are further grouped into three sub-layers. The first sub-layer consists of a toolbox module, which provides some basic facilities like runtime environment management, input database, memory management, low level I/O facilities like interfaces to HDF5 [15]. The toolbox module also provide two runtime systems, one for undirected graph communications, the other for data driven computing on directed graphs. The second sub-layer consists of three modules, the Patch Hierarchy and Patch Data module for mesh and field data storage, and the Communication module for data communications. The third sub-layer consists of the Mesh Adaptive refinement module, Load Balancing module and Math Operation module.

The middle layer contains the modules for the common numerical algorithms shared by many applications including mathematical operations on matrix and vectors, time integration schemes, multi-physics coupling support, mesh generators, some assistant toolkits, and so on.

The top layer provides programming interfaces. This layer is a virtual layer consisting of C++ interfaces for parallel programming. On the top of this layer, users can write serial and numerical subroutines for physical models, parameters, numerical schemes like FVM and FEM, special algorithms, and so on. These subroutines constitute the application program.

JAUMIN is coded in C++, MPI and OpenMP, and can be installed on personal computer, cluster, and massively parallel computer. Through carefully designed user interfaces, parallel implementation details are hidden from application developers. The goal is to support domain experts to code sequentially on desktop computers yet able run their applications on parallel machines.

Fig. 2. The software architecture of JAUMIN.

B. Data structures

A patch is a part of the whole mesh with mesh data and field data stored on it. It is the basic scheduling unit of the framework. The meshes are organized in hierarchical form. As Fig. 3 shows, the whole mesh is firstly partitioned into four sub-mesh and distributed to four processors. In each processor, the sub-mesh is further divided into patches.

Fig. 3. Mesh partition process in JAUMIN.
The union of all the patches is called patch level or simply level. In multi-physics simulation, the computational domain can be described by several levels, with each level corresponding to a physical process. In parallel simulation, patches are distributed among processes, with each process assigned at least one patch. Within a process, multiple threads are scheduled to fetch the local patches and do computations on them.

In the computation based on domain decomposition method, ghost mesh is a way of caching remote data so as to ensure that the simulation be executed correctly in parallel. JAUMIN support “NODE” and “EDGE” ghost mesh in two-dimensional cases, as shown in Fig. 4. And in three-dimensional cases, "FACE" ghost mesh is supported in addition.

From a patch, users can access the geometry data, the connectivity of mesh entities, and entity subsets defined by the user. JAUMIN supports arbitrary types of element shape, such as line, triangle, quadrangle in 2D and tetrahedron, hexahedron and pyramids in 3D. Besides, JAUMIN supports hybrid mesh and user customized element shape.

In the field data management, JAUMIN support physics variable locating on the center of the element, vertex, edge, face, and the boundary of patch. JAUMIN provides efficient implementation for these field data. If the field data type provided by JAUMIN cannot meet the application computational requirements, users can create new field data types by inheriting the base class.

Field data structure plays an important role for efficient computing. To improve the computational efficiency, a array-based unified storage scheme is developed based on patch. This scheme supports scalar, vector, matrix, and particle variables and is independent of all the field data type.

C. Communication patterns

In JAUMIN, two types of communication patterns are supported, one is communication based on undirected graphs, the other is data driven computing on directed graphs.

Communication based on undirected graphs appears in four cases. The first is filling ghost mesh with internal data of adjacent patches, which is a common operation in the simulations based on FDM and FVM. The second case is point by point reduction of patch boundary data, such as force assembling in FEM computation of structural mechanics. The third case is sparse matrix and vector assembling, as well as some matrix vector operations. The last case is data transportation between two levels, which is usually used in the dynamic load balancing and multi-physics simulations. In all these cases, an undirected graph is used to describe the data dependencies and neighboring relationships between patches. In the undirected graph, a node represents a patch, and an edge represents a communication relationship between two patches.

Data driven computing on directed graph is mainly used in particle transportation simulations. In these simulations, there are dependencies between cells. A cell cannot start its computation until all its dependent cells finishes computations. And if a cell has remote dependent cells, the computing result of those cells should send to the ghost mesh of that cell before it can start computing. JAUMIN framework supports dynamic scheduling of cells for computation, and overlapping communications between pairs of depending cells.

Both types of communication patterns are patch centric. That is to say, the computation and communication take the patch as the smallest unit in scheduling, though a patch can be scheduled multiple times in a single pass of data driven computation.

D. Load balance

Load balancing strategies are crucial for the run-time performance of JAUMIN programs. A patch-based dynamic load balancing strategy is realized in JAUMIN, some mature load balancing methods are integrated in this strategy, such as the geometrical bisection methods [16], space filling curves [16] and greedy methods, etc.

Fig. 5 depicts the dynamic load balancing procedure in JAUMIN. In a multi-physics simulation, JAUMIN achieves load balancing individually in each physical computation. In each level, which corresponds to a physical model, the patches are distributed among processors according to their computation loads. In the estimation of workload, we provide programming interface for user to set the workload for a patch. If a patch is too large, it will be divided into small patches. JAUMIN support graph based and geometry based partitioning by calling external packages like Metis and Zoltan.

E. Numerical algorithms

Various solvers for linearly algebraic and eigenvalue problems are integrated into JAUMIN. Efficient iterative solvers are designed for linear systems and some well known libraries are integrated such as Hypre [17] and PETSc [18]. To support these solvers, various mathematical operators for matrices and vectors are also implemented.

The time integration module provides the workflow for time steps on a level. The workflow contains the setup
phase for the initialization of field data, the time integration phase for the discrete stencils, the post-processing phase for memory management, and so on; both explicit and implicit stencil are supported. For implicit stencil, a linear/nonlinear discrete system should be solved using the solver modules.

F. Application programming interface

There are two main features of JAUMIN’s application programming interfaces. One of the feature is patch centric. Applications should provide subroutines that take as computing object. The second feature is component based computing flow. JAUMIN provides a variety of component prototype. A component prototype performs some particular functions, such as data communication, memory allocation, data initialization, etc. A component prototype can be a patch subroutine to form a concrete component, which is the building block of the whole computing flow. The users can build application codes according to the programming workflow shown in Fig. 6.

![Fig. 6. Work flow for development of application based on JAUMIN](image)

The programming workflow consists of three steps. Firstly, the application experts develop standard serial code based on a patch and verify its correctness. Then, components can be configured by combining these serial subroutines with appropriate component prototype using strategy design pattern [19]. Finally, these components are assembled into computational flow. A high performance application is then developed.

In this programming workflow, application experts do not even touch any parallel programming. Most of their work is to develop the serial code and make sure it work correctly. The parallelization is done automatically by component prototypes from JAUMIN. That is to say, User just program on a patch. JAUMIN deal with the data communication between patches.

III. APPLICATIONS

Currently, we have 10 application programs developed on JAUMIN, as shown in Fig. 7. The application domains covered by these programs include structural mechanics, impact dynamics, neutron transport, electromagnetic analysis, etc. We will introduce some selected applications in detail.

The application groups benefit a lot from both the user-friendly programming interfaces and the high performance computational capability provided by JAUMIN. Fig. 8 shows the overview of the development progress and the computation capability of four typical software packages based on JAUMIN. PANDA-STAVIB and PANDA-IMPACT, for example, were developed from scratch, and scaled up to thousands of CPU cores. Up to 74 thousand lines of FEM code were written for PANDA-STAVIB within only 9 months by 5 developers. The Sn code JSNT-U and the FVM code CALE were refactored from the legacy serial codes within three months only. They both obtained more than 1000 times acceleration in computational performance.

A. structural static and vibration simulations

PANDA-STAVIB and PANDA-IMPACT are two important components of PANDA (Platform for Parallel Adaptive Nonlinear Dynamic-and-static Analysis). PANDA, which built on JAUMIN, is a high performance CAE platform oriented to the large equipment manufacture. PANDA-STAVIB is focus on structural static and vibration analysis. Fig. 9 shows a structural static simulation of a part of a machine by PANDA-STAVIB. A hybrid mesh with mixed tetrahedron and hexahedron with hundreds millions of DOFs is used in the simulations. PCG solver is used to solve the linear system.

![Fig. 7. The applications based on JAUMIN.](image)

![Fig. 8. The repaid development of high performance application program based on JAUMIN.](image)

![Fig. 9. Simulation of a machine part. (a) the patches, (b) the deformation.](image)
It is observed from TABLE I that, we achieved 45% strong scalability on 2048 CPU cores.

TABLE I. STRONG SCALABILITY TESTING OF PANDA-STAIVIB IN SIMULATING A MACHINE PART.

<table>
<thead>
<tr>
<th>Number of CPU cores</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time of solution (sec.)</td>
<td>748</td>
<td>414</td>
<td>307</td>
<td>208</td>
</tr>
<tr>
<td>Parallel efficiency</td>
<td>100%</td>
<td>90.3%</td>
<td>80.9%</td>
<td>45.0%</td>
</tr>
</tbody>
</table>

Fig. 10 shows the capability of structural modal analysis built in PANDA-STAIVIB[20]. The CAD model, coming from the target part of the Inertial Confinement Fusion system, includes a large number of sharp corners and circle tubes with the dimension range from 1 centimeter to more than 100 centimeter. It is very difficult to generate a large scale mesh with high quality for this geometric model. A hybrid mesh with mixed tetrahedron and hexahedron is finally achieved in order to perform the simulation. Jacobi-Davison algorithm is adopted to search the eigenvalues iteratively for the assembled sparse system by FEM.

We have done some performance tests on Tianhe-2. As shown in TABLE II, relative to 1024 CPU cores, the parallel efficiency on 8192 CPU cores reaches 62%.

TABLE II. STRONG SCALABILITY TESTING OF PANDA-STAIVIB IN VIBRATION ANALYSIS.

<table>
<thead>
<tr>
<th>Number of CPU cores</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time of solution (sec.)</td>
<td>93671</td>
<td>59007</td>
<td>32316</td>
<td>18867</td>
</tr>
<tr>
<td>Parallel efficiency</td>
<td>100%</td>
<td>79.4%</td>
<td>72.5%</td>
<td>62.1%</td>
</tr>
</tbody>
</table>

B. structure impact dynamic simulations

PANDA-IMPACT is an impact dynamics software package for structural mechanics analysis based on JAUMIN. It has been well tested to scale up to thousands of CPU cores on HPC systems. Fig. 11 illustrates a numerical simulation on a simplified weapon impacting the ground on 256 CPU cores with 20 million DOFs. Again, a hybrid mesh mixed with tetrahedron, wedge and hexahedron is used for the simulation. It should be noted that benefiting from the support of the optimized searching algorithms for geometric entities, PANDA-IMPACT obtained an ultrahigh efficient in this dynamic contact simulation. The numerical results agree well with those of commercial software.

Fig. 11. An impact dynamics simulation for a simplified weapon impacting the ground. (a) the effective stress; (b) comparison of displacements at the tip of the weapon between PANDA-IMPACT and LS-DYNA.

C. Structural mechanics simulations of dam

Saptis is another successful project of JAUMIN to port a serial program to a parallel one. The code is developed by China institute of water resources and hydropower research and is focus on the mechanical performance of the whole life cycle of dams. Now the code has been built on JAUMIN and achieved several thousands of times speed-up than before. Fig. 12 shows a simulation for a real-world dam. The unstructured mesh model is illustrated in (a) and (b), distributed patches in (c) and simulation results of geological subsidence induced by filling the reservoir in (d) where the color shows the displacement of the dam and the ground surface after the reservoir filled with water.

Fig. 12. Simulation result of geological subsidence induced by filling the reservoir

D. Neutron transport simulations

JSINT-U is a neutron transport simulation software based on discrete ordinate method. It is used in nuclear reactor core criticality simulations. Fig. 13 shows the result of our
simulation of a FBR nuclear reactor core on Tianhe-2, using more than ten thousands of CPU cores.

![Diagram](a) ![Diagram](b) ![Diagram](c) ![Diagram](d)

Fig. 13. Criticality simulation of a small FBR nuclear reactor core by JSNT-U. (a): configuration of the reactor. (b): patches. (c): angular flux in the x=0 and y=0 crosssection. (d): angular flux in the center plane cutting the z-axis.

In this simulation, there are 16 energy groups and 24 sweeping angles on 3D tetrahedral meshes. We perform the weak scalability test by fixing roughly the workload on each processor. TABLE III. shows the solution time and parallel efficiency.

TABLE III. WEAK SCALABILITY TESTING OF JSNT-U.

<table>
<thead>
<tr>
<th>Number of CPU cores</th>
<th>24</th>
<th>192</th>
<th>1536</th>
<th>12288</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Elements</td>
<td>414,790</td>
<td>3,010,673</td>
<td>23,257,296</td>
<td>186,658,266</td>
</tr>
<tr>
<td>Time of solution/sec.</td>
<td>36.76</td>
<td>36.59</td>
<td>50.48</td>
<td>85.17</td>
</tr>
<tr>
<td>Parallel efficiency</td>
<td>100%</td>
<td>100.47%</td>
<td>72.82%</td>
<td>43.16%</td>
</tr>
</tbody>
</table>

IV. CONCLUSIONS

In this paper, the prototype of JAUMIN is presented, and some technical details are discussed. The framework highlights itself with its ability to perform halo-exchange based computing, digraph based data-driven sweeping, contact-impact simulation, as well as linear and eigen solver support. Also, its layered and modular design paradigm ensures that it is extensible in functions and adaptable to fast changing hardware technologies. JAUMIN provides with component based application user interfaces, which hide the parallel implementation details from application developers. Several applications, including structural analysis, impact dynamics and neutron transport, are presented. They all feature themselves with fast development cycle and scalability to thousands or tens of thousands of CPU cores.

Acknowledgment

This work was under the auspices of the National Natural Science Foundation of China (Grant Nos. 11372049, 91430218, 61370067). Thanks for the many contributions from members of the Software Center for High Performance Numerical Simulation of CAEP. Thanks for the requirements and suggestions from JAUMIN users. They are the most important driver for us to improve JAUMIN. The visualization results are produced by TeraVAP, which is a data visualizer for large scale science and engineering computing.

References