Soleil-X: Turbulence, Particles, and Radiation in the Regent Programming Language

Hilario Torres
Department of Mechanical Engineering
Stanford University
Stanford, USA
hctorres@stanford.edu

Manolis Papadakis
Computer Science Department
Stanford University
Stanford, USA
mpapadak@cs.stanford.edu

Lluís Jofre
Center for Turbulence Research
Stanford University
Stanford, USA
jofre@stanford.edu

Wonchan Lee
Computer Science Department
Stanford University
Stanford, USA
wonchan@cs.stanford.edu

Alex Aiken
Computer Science Department
Stanford University
Stanford, USA
aiken@cs.stanford.edu

Gianluca Iaccarino
Department of Mechanical Engineering
Stanford University
Stanford, USA
jops@stanford.edu

***Abstract*—The Predictive Science Academic Alliance Program (PSAAP) II at Stanford University is developing an Exascale-ready multi-physics solver to investigate particle-laden turbulent flows in a radiation environment for solar energy receiver applications. To simulate the proposed concentrated particle-based receiver design three distinct but coupled physical phenomena must be modeled: fluid flows, Lagrangian particle dynamics, and the transport of thermal radiation. Therefore, three different physics solvers (fluid, particles, and radiation) must run concurrently with significant cross-communication in an integrated multi-physics simulation. However, each solver uses substantially different algorithms and data access patterns. Coordinating the overall data communication, computational load balancing, and scaling these different physics solvers together on modern massively parallel, heterogeneous high performance computing systems presents several major challenges. We have adopted the Legion programming system, via the Regent programming language, and its task parallel programming model to address these challenges. Our multi-physics solver Soleil-X is written entirely in the high level Regent programming language and is one of the largest and most complex applications written in Regent to date. At this workshop we will give an overview of the software architecture of Soleil-X as well as discuss how our multi-physics solver was designed to use the task parallel programming model provided by Legion. We will also discuss the development experience, scaling, performance, portability, and multi-physics simulation results.**

***Index Terms*—Multi-Physics, Flow Solver, Regent, Legion, Task Based Parallelism**

I. Introduction

The interaction between particle-laden turbulence and thermal radiation plays a critical role in several applications. The most prevalent examples include the interaction of gases, soot, and thermal radiation in combustion systems as well as the study of fuel particulates in solid rocket booster operation. Additionally, concentrated solar energy particle-based receivers are an emerging application of interest that involves these same physical interactions. These types of systems are the focus of the Predictive Science Academic Alliance Program (PSAAP) II at Stanford University [1], a collaborative effort between the Mechanical Engineering and Computer Science departments.

The goal of the program is to run high-fidelity predictive simulations of a particular concentrated solar energy receiver design. The effective use of modern high performance computing (HPC) systems is driven by the computational expense of simulating this device at the required scale. However, efficiently parallelizing and load balancing the drastically different physics solvers and algorithms that are required to run these simulations presents a major challenge. Additionally, doing so in a performance portable way is especially difficult due to the diverse set of heterogeneous system architectures that are prevalent among many HPC systems today. Addressing these challenges with an MPI+X type approach would put a major burden on the application programmers, requiring them to be experts in both the physics and the development of HPC codes in the MPI+X framework.

It is for this purpose that the second major goal of the PSAAP project is focused on developing and using an alternative to MPI+X, the Legion/Regent Programming system [4], [5], for multi-physics simulations in present and future HPC environments. The Legion task-based runtime provides an environment to dynamically load balance with minimal input from the developer/user. Additionally the sequential appearance of the Regent source code, with the parallelization automatically determined at runtime via task/data dependencies, allows the physics domain experts on the project to focus on the physical models and numerical algorithms instead of the parallel performance. This enables them to make substantial contributions to the solver after only a short on-boarding period and avoids the need for them to become experts in MPI+X programming altogether. These advantages are gained without sacrificing scalability. Additionally, using Regent also

This work was funded by the United States Department of Energy's National Nuclear Security Administration under the Predictive Science Academic Alliance Program II at Stanford University, Grant DE-NA-0002373.

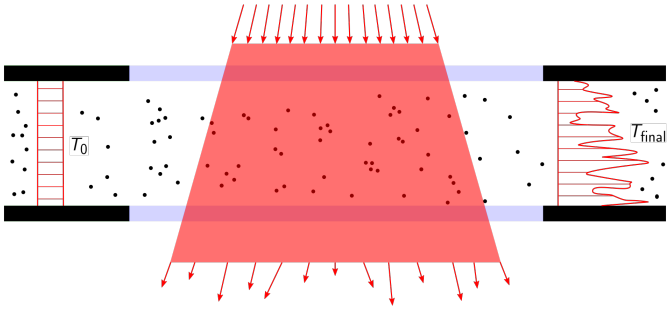

Fig. 1. Cross sectional schematic of a volumetric concentrated solar energy receiver. The black and light blue areas bounding the domain represent opaque and transparent walls, respectively. The flow is from left to right. A uniform initial temperature profile ,T_0, is shown near the inlet to the irradiated section. As the solid particles, represented by the black circles, flow through the irradiated section, represented by the transparent red trapezoid, they absorb heat. As the particles are advected down stream they exchange heat with the fluid and result in the higher temperature profile, T_{final}, at the outlet.

reduces the number of lines and improves the readability and maintainability of the code when compared to similar solvers that use the MPI+X framework.

II. Concentrated Solar Energy Receiver

The schematic in Figure 1 shows the apparatus that has been the focus of a comprehensive experimental and computational study as a part of PSAAP at Stanford. It depicts a fully developed particle laden turbulent duct flow entering the domain of interest, a 4×4 cm square cross section duct, at the left side of the schematic. The fluid is considered to be an ideal gas that is transparent to thermal radiation over the optical depth of the duct. Room temperature air is supplied at the inlet in the simulations and experiments discussed here. The particles are considered to be solid spheres that exchange momentum and energy with the flow and whose radiative absorption and scattering properties are known. In the current set of simulations and experiments the particles are a polydisperse mixture of nickel particles with a mean diameter of 12 microns. The flow and solid particles reach a part of the test section where the walls are not opaque to thermal radiation. In the laboratory-scale experiment that has been built at Stanford this section of the duct is made of glass. This transparent section is exposed to large amounts of thermal radiation that is supplied from an external source. In a full scale system this would be concentrated solar radiation and in the lab scale experiment the test section is irradiated with up to 3 kW by a laser diode array. As the solid particles pass though this irradiated section they absorb thermal radiation and increase in temperature. As stated previously, the working fluid is essentially transparent to the thermal radiation over the width of the duct. However, the particles exchange heat with the fluid as they both flow downstream. This results in a high temperature air-particle mixture downstream of the irradiated section. A short distance after the heated section the particles can be separated from the flow and sent back though the energy receiver. The high temperature working fluid can then be used for power generation.

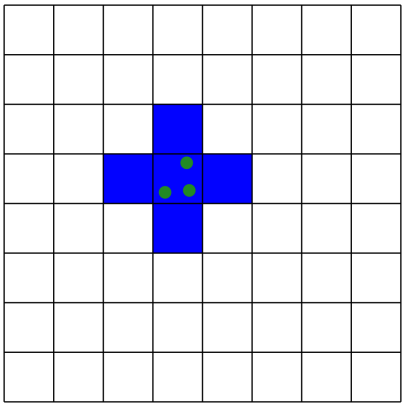

Fig. 2. Fluid solver data access schematic. When the stencil is applied to the fluid cell it's orthogonal neighbors and the particles contained within the center cell must be accessed.

III. Physics Solver Algorithms and Data Access

Our solver simulates this system by doing an explicit time integration of the governing equations for each of the relevant physics. The fluid phase conservation equations for mass, momentum, and energy are solved on an Eulerian mesh. The position, momentum, and energy equations are integrated for each particle. The radiative transfer equation is solved on an Eulerian mesh using the discrete ordinates method. Each of the physics solvers store and access data in very different ways. Additionally, some data does need to be exchanged between the physics solvers due to the coupling between them. The full set of governing equations will not be discussed here due to space limitations but can be found in previous papers [2], [3]. However, a high level discussion of the solver algorithms and how they are coupled is provided below.

At each time step the fluid solver loops over the fluid cells and applies a discretization stencil to compute the convection and diffusion fluxes needed to advance the equations of fluid motion. This step requires read access to several fields for all cells within the stencil and write access to the cell at the center of the stencil. Additionally the particles inside of the center fluid cell contribute to the fluid momentum and energy via drag forces and heat fluxes, so several of the fields associated with the particles inside of the cell must be read by the fluid solver. Figure 2 represents the data that is accessed when applying the stencil to a fluid cell.

The particle solver requires looping over all of the particles with read and write access to several of the particle fields. The particle solver interacts directly with both the fluid and radiation solvers. The fluid contributes to the energy and momentum of the particles, which requires reading data from the fluid cells that surround the particle. This process is depicted in Figure 3 and requires searching for which cell each particle is in. The particles also move (due to their own inertia as well as drag forces by the underlying fluid) as a result of each time-step, so that this mapping of fluid to particles and *vice versa* must be done each time-step. An additional challenge caused by the particles' motion is that they must

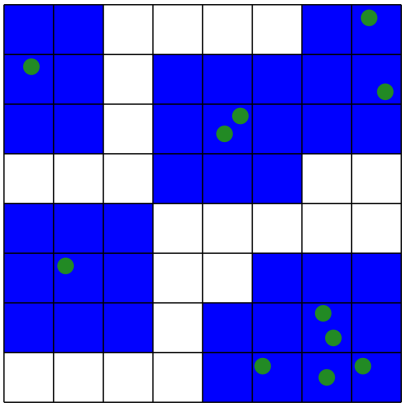

Fig. 3. Particle solver data access schematic. The particles, represented by the green circles, and the blue fluid cells that contain the particles are accessed by the particle solver.

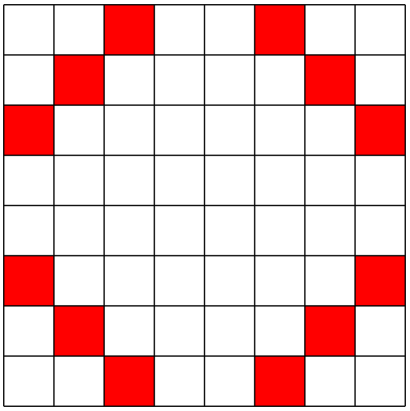

Fig. 4. Radiation solver data access for third step of a DOM sweep. The radiation solver is reading from and writing to the red cells.

be exchanged between sub-regions of the domain when the domain is decomposed for parallel processing.

The radiation solver uses the discrete ordinates method (DOM) [6] to solve for the radiation field on an Eulerian mesh, which can be significantly coarser than the Eulerian mesh used for the fluid because of vastly different resolution requirements. Before the DOM solver is able to run, the radiative absorption and scattering properties must be defined on this Eulerian mesh. These properties are directly dependent on the number of particles in each cell and their diameter, requiring read access to the particles located within each radiation cell. Then the DOM requires "spherical sweeps" over the radiation mesh starting from the corners of the domain. A schematic of the third step of this sweep is shown in Figure 4.

IV. Solver Architecture

The highly coupled nature of the three physics solvers outlined in the previous section make it difficult to efficiently parallelize this application manually in the MPI+X framework, which is one of the main reasons the decision was made to use the task parallel programming model provided by Regent.

```
task ComputeKineticEnergy(cells : region(ispace(int3d),Cell))
where
  reads (cells.Rho, cells.Velocity)
  writes(cells.Kinetic_Energy)
do
  for cell in cells do
    cell.Kinetic_Energy = 0.5 * cell.Rho * cell.Velocity^2.0
  end
end
```

Fig. 5. Example of a task definition in Soleil-X. The fields with read and write privileges, which are used to parallelize tasks, are highlighted in blue and red respectivly.

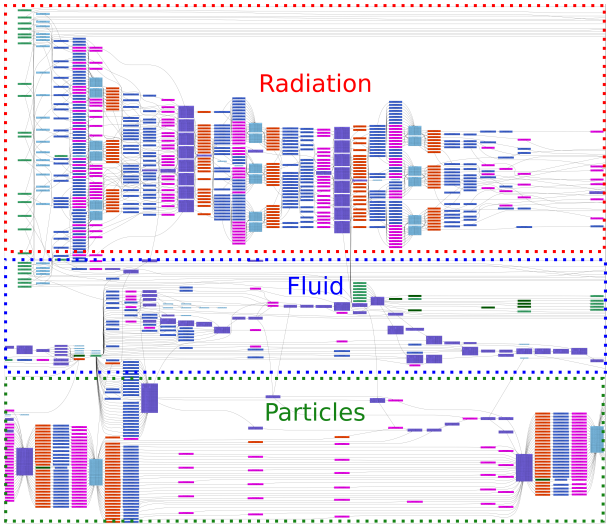

Fig. 6. Data flow task graph produced by Legion Spy tool for one time step in Soleil-X. The dashed boxes highlight the tasks associated with each physics solver.

We chose to use the Regent programming language instead of the Legion C++ API so that the physics domain experts could develop the code with only a high level understanding of how the Legion programming system works. This development method was largely successful, allowing the physics domain experts to focus on the physics after learning only a few high level concepts about the programming model and Regent syntax. This allowed them to write what was essentially a serial code that could be parallelized at runtime to run on HPC systems. The tasks are automatically parallelized based on the order in which they are launched and the privilege declarations for fields that tasks operate on. Figure 5 shows an example of how the field privileges for a task in Soleil-X are declared and used. The fields with read and write access are highlighted in blue and red respectively.

A portion of the data flow graph for one time-step of the solver, generated by the Legion Spy tool, is shown in Figure 6. The colored boxes represent tasks and the solid lines represent data dependencies. The dashed lines were added to draw attention to the fact that each physics solver has many data dependencies between the tasks within it and also several with the other physics solvers. The solid lines crossing the dashed lines represent the flow of data between the different

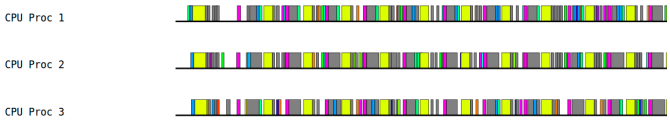

Fig. 7. Profile of Soleil-X generated by the Legion profiling tool. The tight packing of tasks and lack of white space indicate good utilization of the machine.

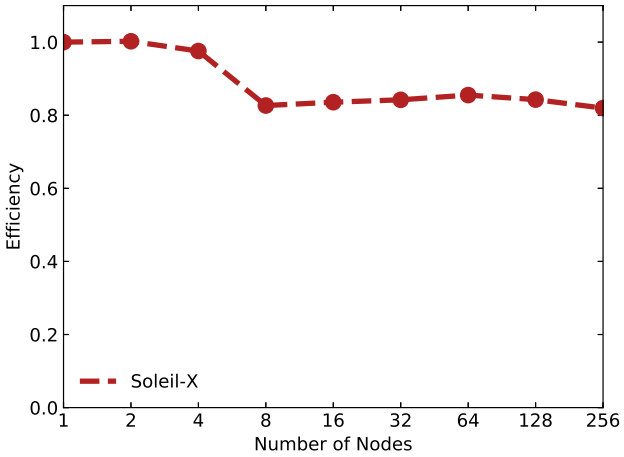

Fig. 8. Full-physics weak scaling of Soleil-X on the Sierra supercomputer.

physics solvers. Figure 7 shows a profile, generated using the Legion profiling tool, of the solver running on a portion of one node. The horizontal axis represents time and the width of the colored blocks represent the duration that a task was running. The lack of gaps between the task blocks indicates that the machine is being effectively utilized.

V. Solver Performance and Portability

We were easily able to port Soleil-X across a range of machines, from single-processor laptops to local clusters at Stanford (Certainty, Sherlock), and to leadership-class machines (Titan, Piz Daint, Lassen, Sierra, Summit). Two features of the Legion platform were crucial in this: the cross-platform nature of the runtime itself, and the Regent compiler's support for a wide range of processor and accelerator architectures through its use of the LLVM compiler library.

After achieving correctness at a small scale we were able to leverage the flexibility of the Legion runtime to scale up to the sizes necessary for our capstone experiments with little additional effort. Figure 8 shows some of the results of our latest weak scaling study.

These tests were performed on the Sierra supercomputer, using just the 4 NVIDIA Tesla V100 GPUs on each node to solve the physics. We ran on a triply periodic domain that we split across our nodes using a straightforward volumetric decomposition, such that each node is assigned 256^3 fluid cells and around 32M particles. These numbers were chosen to be close to the GPU memory limit on a node. We ran our solver once for each node count, completing 50 timesteps, and solving full physics on each timestep (we disregard 5 warm-up and 5 ramp-down timesteps in our results). The 1-node case completes in 125 seconds. We chose to use a simplified radiation model instead of DOM for this study, as the DOM method has inherent scalability issues that we did not want to conflate with the scalability of the main solver code.

Overall, our solver demonstrated good weak scaling behavior up to 256 nodes. What appears like a drop in performance from 1 to 8 nodes can be attributed to an increase in required communication: When our fluid domain is not split along some dimension (e.g. along the x dimension on a 1x2x2 tiling, which is used for the 4-node case), there is no need to trade ghost cells on that dimension between iterations. Starting at 8 nodes (where we follow a 2x2x2 tiling) every node has the maximum number of neighbors (two along each dimension), and thus performance stabilizes.

Unfortunately, at this time we cannot offer a baseline performance comparison of Legion versus MPI+X for our application. Due to the complexity of our application, we did not have enough resources to devote to doing a proper re-implementation of Soleil-X on a different programming system so that we could perform a direct comparison.

VI. Conclusions

The PSAAP center at Stanford University is a collaborative effort between mechanical engineers and computer scientists with a dual research goal: (i) understanding the coupled multi-physics phenomena involved in a novel-type of solar energy receiver using large scale simulations, and (ii) developing the computer science framework required to build the multi-physics solver and achieve performance and portability on leadership-class supercomputers. To address these goals we have developed a multi-physics solver written entirely in the Regent programming language, Soleil-X. The various physics solvers and numerical algorithms that are implemented in Soleil-X were described. The performance and portability of the code was also discussed. Our application demonstrates the use of an alternative to MPI+X, the Legion programming system via the Regent programming language, for a complex application that is being used to run simulations at scale on leadership-class HPC systems.

References

[1] Exascale Computing Engineering Center, "Predictive Science Academic Alliance Program (PSAAP) II," Stanford University, 2019.
[2] H. Pouransari, and A. Mani, "Effects of Preferential Concentration on Heat Transfer in Particle-Based Solar Receivers," Journal of Solar Engineering, vol. 139, 2017.
[3] L. Jofre, G. Geraci, H. R. Fairbanks, A. Doostan, and G. Iaccarino, "Multi-fidelity uncertainty quantification of irradiated particle-laden turbulence," CTR Annual Research Briefs, pp. 21-34, 2017.
[4] E. Slaughter, W. Lee, S. Treichler, M. Bauer, A. Aiken, "Regent: A High-Productivity Programming Language for HPC with Logical Regions," SC 15, November 15 - 20, 2015, Austin, TX, USA.
[5] M. Bauer, S. Treichler, E. Slaughter, A. Aiken, "Legion: Expressing Locality and Independence with Logical Regions," SC12, November 10-16, 2012, Salt Lake City, Utah, USA.
[6] M. Modest, "The Method of Discrete Ordinates" in Radiative Heat Transfer, Oxford, UK, Academic Press, 2013, ch. 17