Parallel CFD with Regent

Nicholas Stegmeier, Marc Charest
Computational Physics Division
Los Alamos National Laboratory

The goal of this project was to explore various new parallel programming strategies that were designed for current and emerging computer architectures. One new strategy is Regent, a research programming language which extracts implicit dataflow parallelism from code written with sequential semantics. Regent is designed to exploit both shared and distributed memory architectures. Here, one- and two-dimensional Godunov-type finite-volume methods were implemented to assess Regent's parallel scalability and ease of use.

Introduction to Regent

Regent is a high-productivity programming language for parallel computing that is based on the idea of **logical regions**.

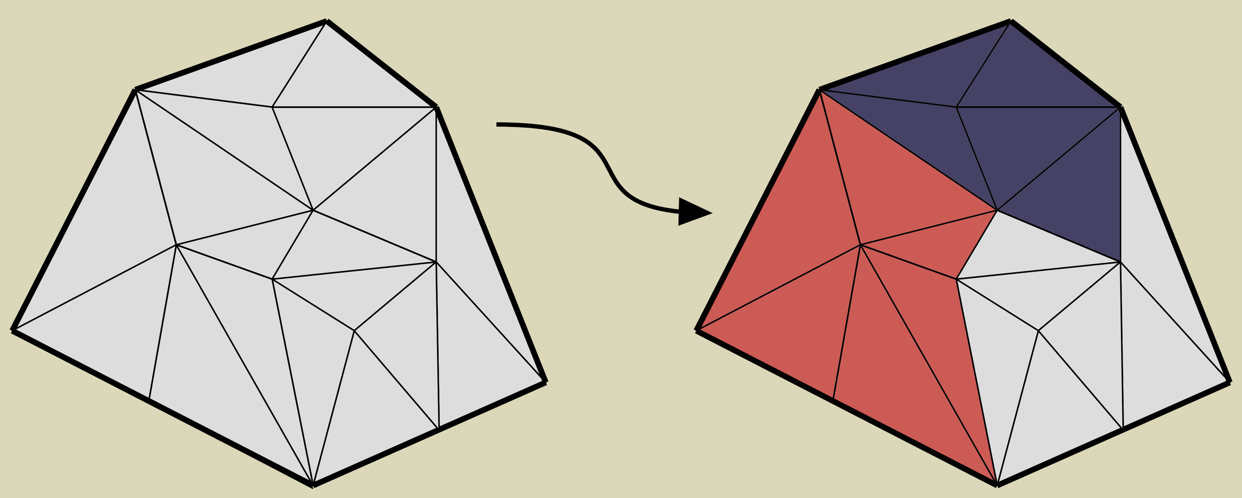
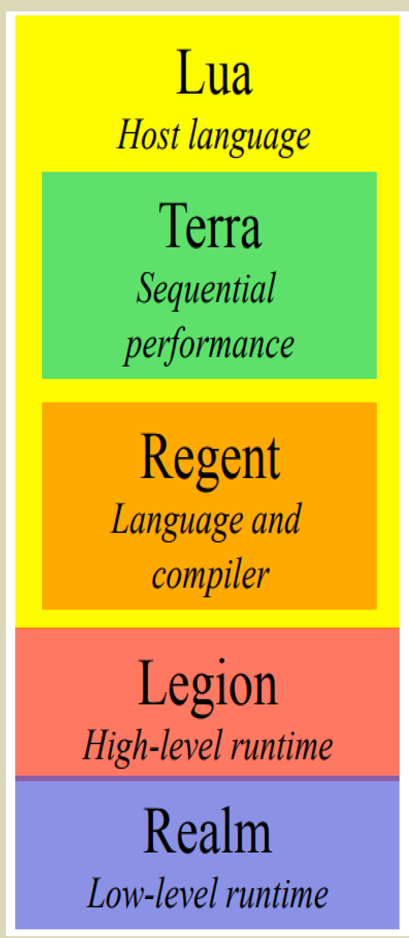


Figure: Region decomposed into different colors.

Design of Regent

- Based on the programming model of Legion
- The runtime handles parallel, asynchronous task scheduling
- Embeded in Lua (a Python-like scripting language)
- Has access to Terra for fast computation



Programming in Regent

- Programs appear to be written sequentially
- Regions of data are defined and decomposed
- Parallelism is extracted automatically from task-region dependency

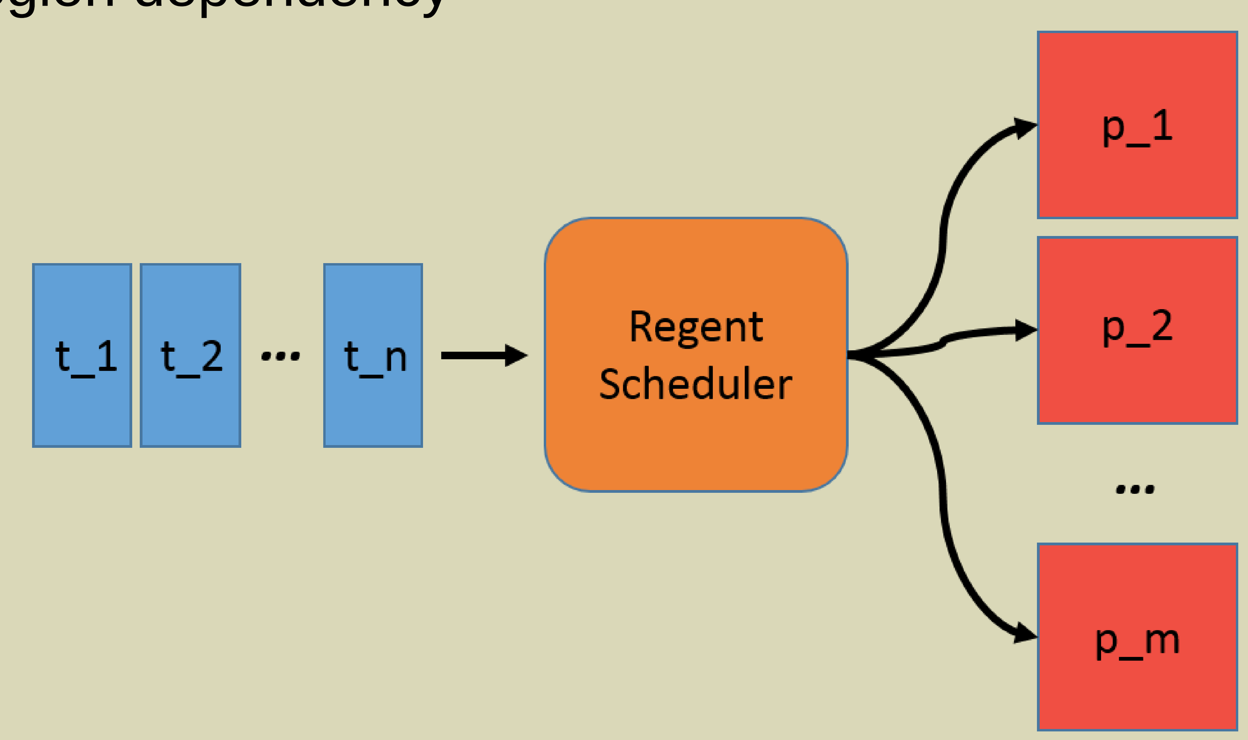


Figure: Regent runtime parallel asynchronous task scheduling.

2D Euler Equations

We solve numerically the 2D Euler equations for gas dynamics:

$$\frac{\partial U}{\partial t} + \nabla \cdot \vec{F} = 0$$

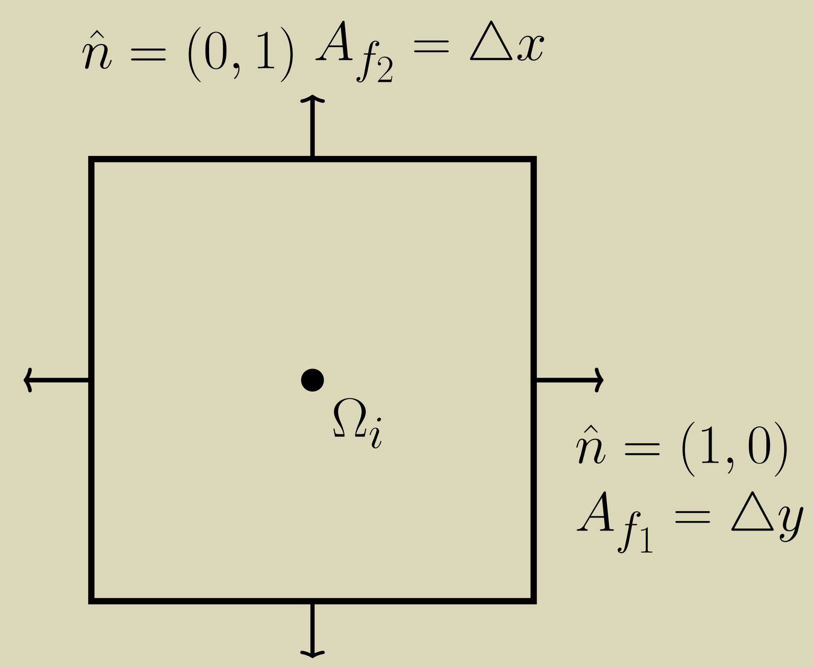
Where the vector U and tensor F are given by

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e_t \end{bmatrix} \quad \vec{F} = \left[\begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho e_t u + up \end{bmatrix} \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho e_t v_+ vp \end{bmatrix} \right]$$

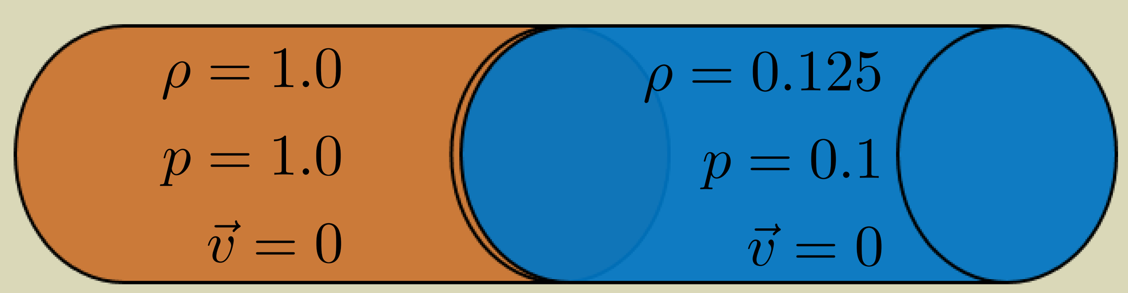
The Euler equations are discretized with Forward Euler in time and the **finite volume method** in space. Integrating the above equations over each control volume gives the following:

$$\frac{\partial \bar{U}}{\partial t} + \frac{1}{V_i} \sum_{f_j \in \Omega_i} (\bar{F} \cdot \vec{n}) = 0$$

Face fluxes are calculated using the Rusanov approximate flux function, and gradient limiters are employed to result in a stable, second order accurate scheme.



Sod Shock Tube Initial Conditions



Sedov Blast-type Problems

- A high specific internal energy is placed into a single computational cell, which translates into a high pressure:

$$p_0 = \frac{(\gamma - 1)\epsilon_0 \rho}{V}$$

Simulation Results

We simulated spatially 1st and 2nd order discretizations of the 1D and 2D Euler equations.

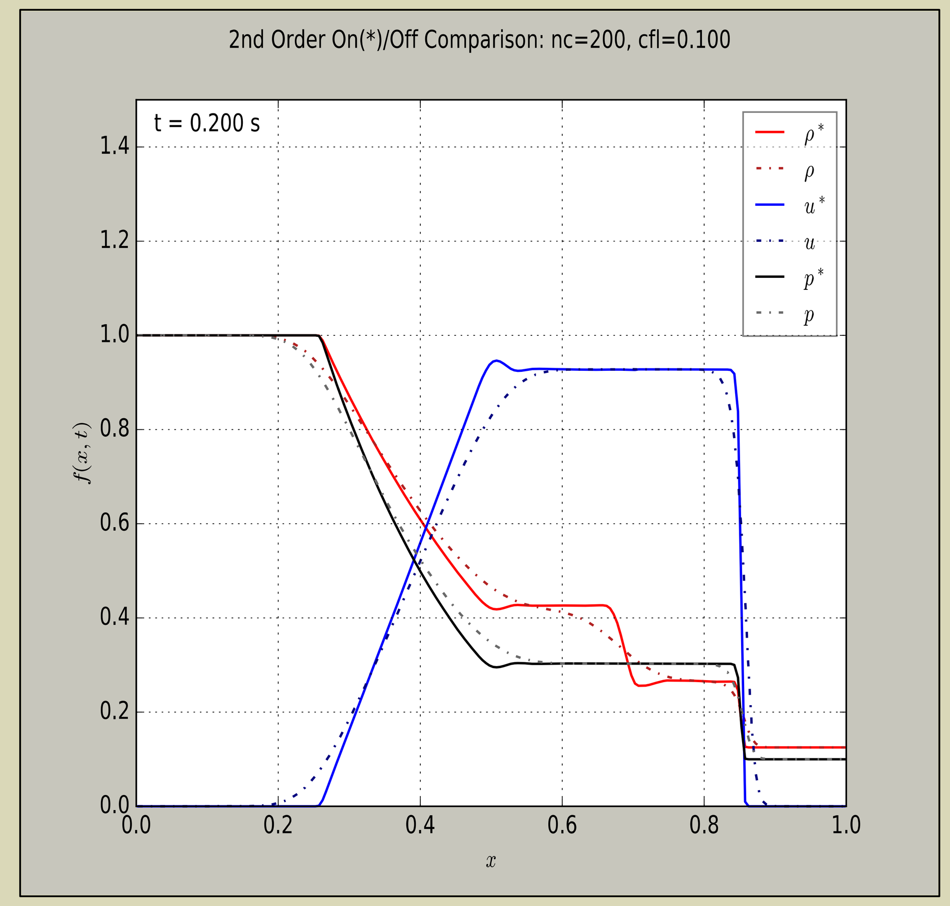


Figure: 2nd order results for the 1D Sod Shock Tube problem.

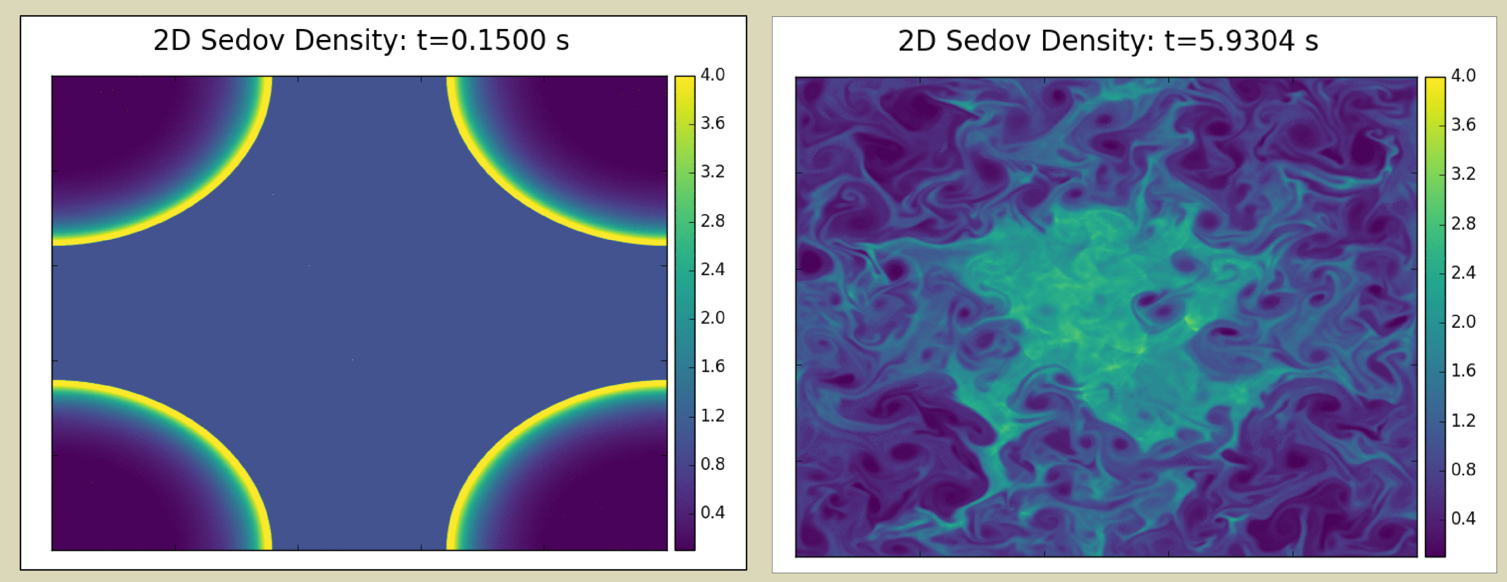
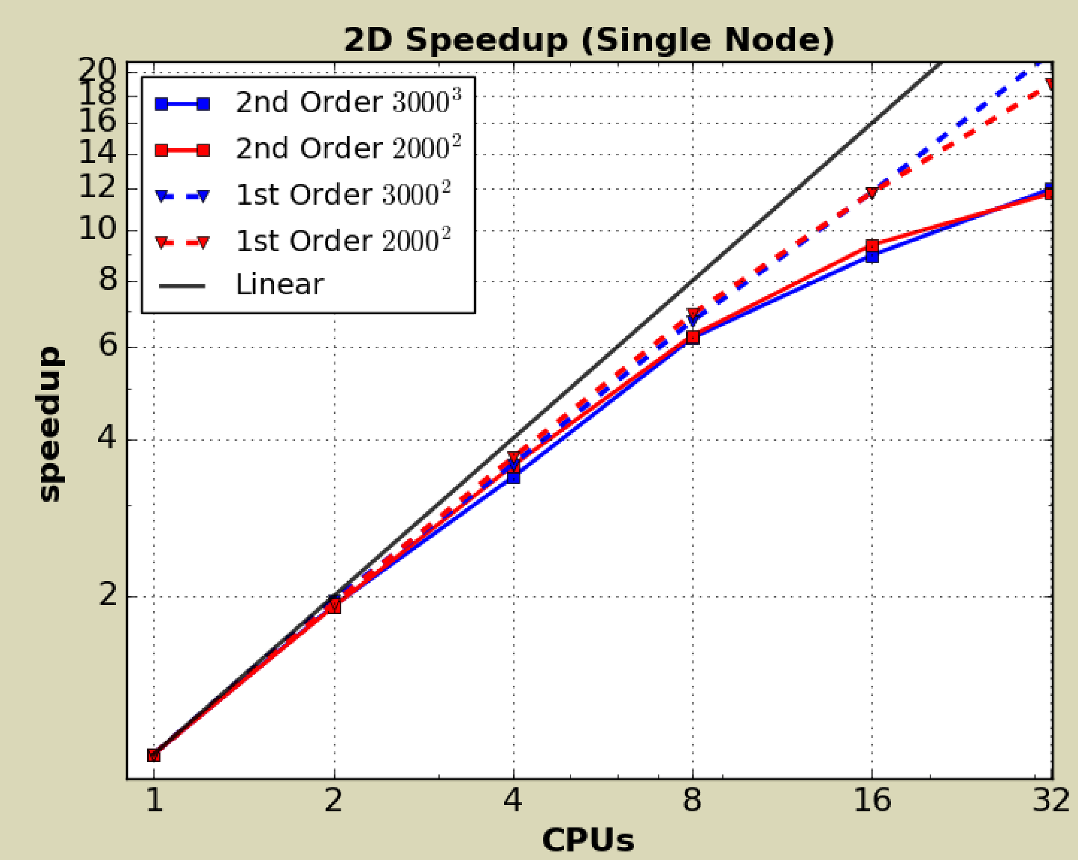


Figure: Time evolution of a multi-blast wave problem.

Strong Scaling Results



Conclusions

- Regent scales well on a single node, although large problem sizes are required.
- Regent provides relatively low-cost parallelism.