High Performance Computing 1b
Parallelization of a 2D Hydro Solver

Remo Hertig
Stephan Radonic

University of Zurich

June 30, 2015

Introduction and physics

Our task was to parallelize an existing C code, originally written by Prof. Romain Teyssier in Fortran, which solves the Euler equations in 2D using a Godunov scheme. The euler equations in conservation form are

$$\partial_t \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ 0 \end{pmatrix} + \nabla \cdot \begin{pmatrix} \rho\mathbf{u} \\ \rho\mathbf{u} \otimes \mathbf{u} + p\mathbf{I} \\ \mathbf{u} \end{pmatrix} = 0 \tag{1}$$

This set of equations describes the flow of a gas basically stating the momentum, mass and energy conservation. A hyperbolic PDE in conservation law form is generally represented as

$$\partial_t \mathbf{U} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0 \tag{2}$$

Discretization on a grid yields

$$\mathbf{U}_i^{n+1} = \mathbf{U}_\mathbf{i}^\mathbf{n} + \frac{\Delta x}{\Delta t}(\mathbf{F}_{\mathbf{i-1/2}} - \mathbf{F}_{\mathbf{i+1/2}}) \tag{3}$$

where $\mathbf{F}_{\mathbf{i\pm1/2}}$ are the fluxes at the cell boundaries, the Godunov scheme uses various approximations for $\mathbf{F}_{\mathbf{i\pm1/2}}$, depending on the specific variation of the method, e.g upwind scheme, lax-friedrich, ...

Parallelization

We implemented a symetric vertical domain decomposition[1] and used nonblocking MPI communication to share ghost cells.
Since the resulting data output is massive (~600Mb for a 10M grid per step) it is currently unpractical to store the results from every step. Therefore we opted for a lower storage resolution. This allows us to implement the "write-to-disk" part in a rather sub-optimal way: MPI_Gather to the master. However the performance impact of this strategy is negligible over a long simulation time.

[1] *Restriction:* $Width = \{n_{proc} k | k \in \mathbb{N}\}$

Exploratory runs

Parallelization: Speedup and Scaling

We compare the average time step durations for a single process up to approximatly 1500 parallel processses for a fixed problem size (in our case 60994 x 120). As observable in the strong scaling graph (ref figure) we get a super linear scaling up to 800 processes. The super linearity of the scaling can be explained with cache usage effects.

Oprimal cache memory usage only works well for rectangular shaped domains (large x small y for parallelization in x direction). We have compared how a fixed sized problem performs with diffrent x/y ratios (see figure xx). We can cleary see that the performance increases with deacrasing y/x size, up to a ratio, where each processes computing domain gets too small and becomes inefficient.

Parallelization: Speedup and Scaling

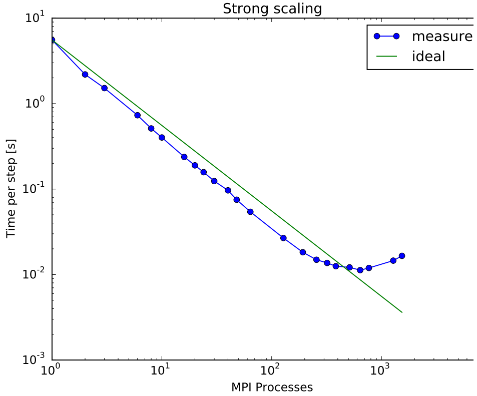


Figure: Strong scaling for a fixed grid size of 69994 $\times$ 120 for 1 to 1536 processes. with the time step decreasing from 4.65 to 0.009

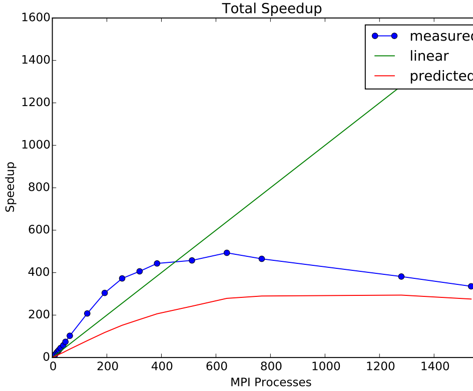


Figure: Performance comparison of a fixed sized grid with varinyg y/x ratio

Parallelization: Speedup and Scaling

...tzututzut

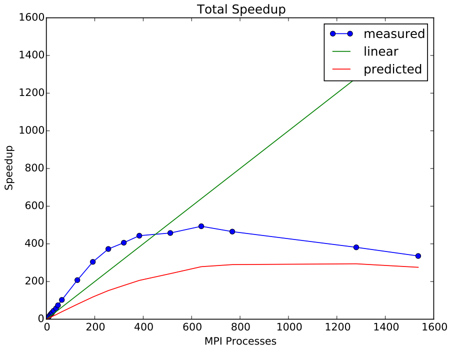
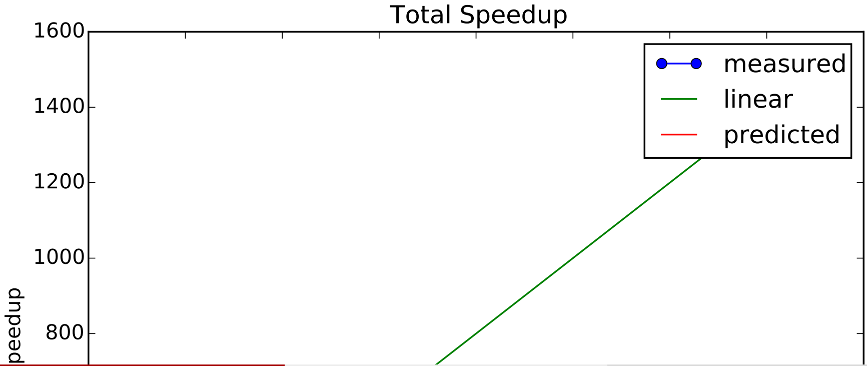


Figure:

Output image

To show the power of parallel processing we wanna show an excerpt from our high resolution image (3060 x 500) at simulation time t=600 seconds (corresponds to the 200'000th time step in our simulation). In order to avoid unnecessary wasting of computing resources, we didn't want to use a larger grid size.


Total Speedup
peedup
measured
linear
predicted