R. Krause, D. Ruprecht

Hybrid Space-Time Parallel Solution of Burgers Equation

ICS Preprint No. 2012-07

11 October 2012

Hybrid Space-Time Parallel Solution of Burger's Equation

Rolf Krause[1] and Daniel Ruprecht[1,2]

Abstract An OpenMP-based shared memory implementation of the Parareal parallel-in-time integration scheme using explicit integrators is combined with a standard MPI-based spatial parallelization of a finite difference method into a hybrid space-time parallel scheme. The capability of this approach to achieve speedups beyond the saturation of the pure space-parallel scheme is demonstrated for the two-dimensional Burgers equation.

1 Introduction

Many applications in high performance computing (HPC) involve the integration of time-dependent partial differential equations (PDEs). Parallelization in space by decomposing the computational domain is by now a standard technique to speed up computations. While this approach can provide good parallel scaling up to a large number of processors, it nevertheless saturates when the subdomains become too small and the time required for exchanging data starts dominating. Regarding the anticipated massive increase of available cores in future HPC systems, additional directions of parallelization are required to further reduce runtimes. This is especially important for time-critical applications like, for example, numerical weather prediction, where there exists a very strict constraint on the total time-to-solution for a forecast in order to be useful.

One possibility for providing such an additional direction of parallelization are parallel-in-time integration schemes. A popular scheme of this type is Parareal, introduced in [2, 6]. It has been applied successfully to a broad range of problems and also undergone thorough analytical investigation. A large number of corresponding references can be found, for example, in [5, 8].

Institute of Computational Science, Università della Svizzera italiana, Via Giuseppe Buffi 13, 6900 Lugano, Switzerland, {rolf.krause,daniel.ruprecht}@usi.ch · Mathematisches Institut, Heinrich-Heine-Universität, Universitätsstrasse 1, 40225 Düsseldorf, Germany

While numerous works exist dealing with different aspects of Parareal in a purely time-parallel approach, there seem to be few studies that address the combination of Parareal with spatial parallelization, in particular with a focus on implementation. First results on combining Parareal with spatial domain decomposition are presented in [7]. While scaling of the algorithm is discussed, no runtimes are reported. In [11, 12], computing times for a pure MPI-based combination of Parareal with spatial domain decomposition for the two-dimensional Navier-Stokes equations are given, but with ambiguous results: Either a pure time-parallel or a pure space-parallel approach performed best, depending on the problem size. In [3], the capability of an pure MPI-based approach to speed up simulations for the 3D Navier-Stokes equations beyond the saturation of the spatial parallelization is shown.

This paper investigates the performance of a combination of a shared memory implementation of Parareal featuring explicit integrators with a MPI-based parallelization of a stencil-based spatial discretization into a hybrid (see [9]) space-time parallel method. The code is an extension of the purely time-parallel, OpenMP-based implementation used in [10]. Using shared memory for Parareal avoids communication of volume data by message passing and should reduce the memory footprint of the code.

2 Algorithm and Implementation

The starting point for Parareal is an initial value problem

$$\mathbf{q}_t = \mathbf{f}(\mathbf{q}), \ \mathbf{q}(0) = \mathbf{q}_0 \in \mathbb{R}^d, \tag{1}$$

where in the present work, the right hand side $\mathbf{f}$ stems from the spatial finite difference discretization of some PDE on a rectangular domain $\Omega \subset \mathbb{R}^2$. The spatial parallelization uses a standard non-overlapping decomposition of the domain, allowing for a distributed computation of $\mathbf{f}(\mathbf{q})$, where every every MPI-process handles the degrees-of-freedom of one subdomain and ghost-cell values are exchanged at the boundaries. As explicit schemes are used here within Parareal, only straightforward evaluations of $\mathbf{f}$ are required.

2.1 Parareal

Parareal allows to parallelize the integration of (1) by combining a number of time-steps into one coarse time-slice and performing an iteration where multiple time-slices are treated concurrently. Let $\mathscr{F}_{\delta t}$ denote an integration scheme of suitable accuracy, using a time-step δt. A second integration scheme is required, typically called $\mathscr{G}_{\Delta t}$, using a time-step $\Delta t \gg \delta t$, which has to be much cheaper in terms of computation time but can also be much less accurate. Denote by

Algorithm 1 Parareal algorithm implemented with OpenMP

```
1: $\mathbf{q}_0^0 = \mathbf{q}_0$, $k := 0$
2: for $i = 0$ to $N_c - 1$ do
3:     $\mathbf{q}_{i+1}^0 = \mathscr{G}_{\Delta t}(\mathbf{q}_i^0, t_{i+1}, t_i)$
4: end for
5: repeat
6:     omp parallel for
7:     for $i = 0$ to $N_c - 1$ do
8:         $\tilde{\mathbf{q}}_{i+1}^k = \mathscr{F}_{\delta t}(\mathbf{q}_i^k, t_{i+1}, t_i)$
9:     end for
10:    omp end parallel for
11:    for $i = 0$ to $N_c - 1$ do
12:        $\mathbf{q}_{i+1}^{k+1} = \mathscr{G}_{\Delta t}(\mathbf{q}_i^{k+1}, t_{i+1}, t_i) + \tilde{\mathbf{q}}_{i+1}^k - \mathscr{G}_{\Delta t}(\mathbf{q}_i^k, t_{i+1}, t_i)$
13:    end for
14:    $k := k + 1$
15: until $k = N_{it}$
```

$$\tilde{\mathbf{q}}_g = \mathscr{G}_{\Delta t}(\mathbf{q}, \tilde{t}, t), \quad \tilde{\mathbf{q}}_f = \mathscr{F}_{\delta t}(\mathbf{q}, \tilde{t}, t) \tag{2}$$

the result of integrating forward in time from an initial value $\mathbf{q}$ at time t to a time $\tilde{t} > t$ using $\mathscr{G}_{\Delta t}$ or $\mathscr{F}_{\delta t}$. Parareal uses $\mathscr{G}_{\Delta t}$ to produce approximate solutions at nodes $(t_i)_{i=0,\dots,N_c}$ of a coarse temporal mesh (lines 2 – 4 in Algorithm 1). These guesses are then used as initial values for running $\mathscr{F}_{\delta t}$ concurrently on all N_c time intervals $[t_i, t_{i+1}]$ (lines 6–10). A correction is then propagated sequentially by another sweep of $\mathscr{G}_{\Delta t}$ (lines 11 – 13). The procedure is iterated and converges towards the solution that would be obtained by running $\mathscr{F}_{\delta t}$ sequentially from t_0 to t_{N_c}. For a detailed explanation and properties of the algorithm we refer to [5] and references therein. Note that an MPI-based implementation of Parareal requires communication of full volume data in line 12, which is avoided by the shared memory parallelization in time used here.

2.1.1 Theoretical Speedup.

For a given time interval $[t_0, t_{N_c}]$, denote by N_f the number of fine steps required to integrate from $t = t_0$ to $t = t_{N_c}$, by τ_c and τ_f the execution time of one single coarse or fine time-step and by N_{it} the number of performed iterations. Further, assume that $\mathscr{G}_{\Delta t}$ always performs one single step, so that N_c is also the number of coarse steps between t_0 and t_{N_c}. The speedup obtainable by Parareal for a given number of processors can be estimated by

$$s(N_p) \approx \frac{1}{(1 + N_{it})\frac{N_c}{N_f}\frac{\tau_c}{\tau_f} + \frac{N_{it}}{N_p}} \leq \frac{N_p}{N_{it}}. \tag{3}$$

Note that the maximum parallel efficiency is bounded by $1/N_{it}$. Recently, a scheme based on a combination of Parareal with spectral deferred corrections named PFASST

Algorithm 2 Ghost-cell exchange between subdomains in hybrid scheme.

```
1: omp barrier
2: omp master
3: MPI_IRECV(Ghostcells $\times$ #threads)
4: MPI_ISEND(Ghostcells $\times$ #threads) and MPI_WAITALL
5: omp end master
6: omp barrier
7: Evaluate $\mathbf{f}$
```

has been introduced in [4]. PFASST significantly relaxes this bound, but still does not allow for optimal parallel efficiency.

2.2 Implementation

For the OpenMP-based parallelization sketched in Algorithm 1 to be efficient, the implementation of the fine propagator has to properly deal with non-uniform memory access (NUMA). This requires correct use of first-touch initialization of buffers allocated in the used modules. Here, inner-module buffers are enlarged by an additional dimension of length #threads and each thread initializes its part of the buffer. This also provides the master thread with access to one large buffer containing ghost-cell values of all threads. Loop parallelizations in the initialization and in Algorithm 1 both use the schedule (static) option to ensure identical thread assignment. Communication between different MPI-processes in Algorithm 1 is hidden in the propagators $\mathscr{G}_{\Delta t}$ and $\mathscr{F}_{\delta t}$. These require repeated evaluation of the right hand side function $\mathbf{f}$ and before each evaluation, a halo of up-to-date ghost cell values is exchanged between neighboring subdomains.

2.2.1 Ghost-Cell Exchange.

MPI is initialized with the MPI_THREAD_FUNNELED option, allowing only the master thread to call MPI routines. As the ghost-cell communication in $\mathscr{F}_{\delta t}$ takes place in the multithreaded part of the code, suitable OpenMP pragmas are used to synchronize threads and ensure compliance with the funneled option. The implementation is sketched in Algorithm 2. The first barrier ensures that the master thread sends up-to-date ghost-cell values, the second that all threads wait until the master thread has finished communicating, so that up-to-date values are used in the evaluation of the right-hand-side function $\mathbf{f}$. The master pragma ensures that only the master thread calls the MPI communication routines, which sends and receives one large buffer containing ghost-cell values for all threads. Note that while the master thread is busy communicating, the other threads are idle. This ”idle threads problem” is one of the drawbacks of the funneled approach pointed out in [9]. As the coarse integrator is run only by the master thread, no thread synchronization is required there. Experiments with options MPI_THREAD_SERIALIZED and MPI_THREAD_MULTIPLE are left for future work. In particular, shared-memory implementations of a pipelined version of Parareal [1] in combination with MPI-based spatial parallelization are of interest here, as pipelining can improve somewhat the provided speedup.

3 Numerical Results

The performance of the hybrid space-time parallel approach is analyzed here for the two-dimensional, nonlinear, viscous Burgers equation

$$u_t + uu_x + uu_y = \nu \Delta u \tag{4}$$

on a domain $[-2,2] \times [-2,2]$ with initial value

$$u(x,y,0) = \sin(2\pi x)\sin(2\pi y), \tag{5}$$

a parameter $\nu = 0.02$, a mesh width $\Delta x = \Delta y = 1/40$ and periodic boundary conditions. A two-dimensional decomposition of the domain into quadratic or rectangular subdomains, depending on the number of MPI-processes, is performed and a cartesian communicator for MPI is used. Parareal uses time-steps $\Delta t = 2 \times 10^{-3}$ and $\delta t = 2 \times 10^{-5}$. For $\mathscr{G}_{\Delta t}$, the spatial discretization uses 3rd-order upwind finite differences for the advection term and 2nd-order centered differences for the Laplacian, while $\mathscr{F}_{\delta t}$ uses a 5th-order upwind stencil for the advection and a 4th-order centered stencil for the Laplacian. Hence, a two-cell wide halo has to be exchanged in the coarse and a three-cell wide halo in the fine propagator. The simulations are run until $T = 0.5$ and $\mathscr{G}_{\Delta t}$ always performs one single step per coarse interval, so the number of restarts of Parareal depends on the number of threads assigned for the temporal parallelization. A forward Euler scheme is used for $\mathscr{G}_{\Delta t}$ and a Runge-Kutta-2 scheme for $\mathscr{F}_{\delta t}$.

To assess accuracy, a reference solution with $\delta t/10$ is computed sequentially. With a fixed number of $N_{\text{it}} = 3$ in Parareal, the relative $\|\cdot\|_\infty$-error of the time-parallel solution is $\varepsilon_{\text{para}} \approx 2.2 \times 10^{-8}$ and of the time-serial solution $\varepsilon_{\text{seq}} \approx 1.8 \times 10^{-8}$, so that both solutions are of comparable accuracy. The coarse integrator run alone results in $\varepsilon_{\text{coarse}} \approx 2.9 \times 10^{-2}$.

The used machine is a cluster consisting of 42 nodes, each containing 2 quad-core AMD Opteron CPUs with 2,700 MHz and 16 GB RAM per node. In the example below, the time parallelization always uses eight threads per node, in order to utilize one full node. The nodes are connected by an INFINIBAND network.

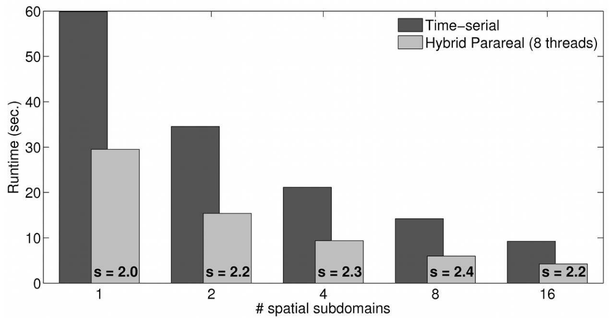

Fig. 1 Runtimes of the space-parallel, time-serial and hybrid space-time-parallel scheme for different numbers of spatial subdomains. The numbers indicate the speedup from the temporal parallelization, that is the runtime of the time-serial solution divided by the time-parallel solution using the same number of spatial subdomains.

3.1 Runtimes and Scaling

Reported runtimes are measured with the MPI_WTIME routine provided by MPI and do not contain I/O operations.

3.1.1 Speedup from Parareal.

With the used parameters, the speedup obtainable by Parareal using eight threads is bounded by $s \leq 2.57$ according to (3). The ratio $\tau_c / \tau_f = 0.35$ has been determined experimentally by running $\mathscr{G}_{\Delta t}$ and $\mathscr{F}_{\delta t}$ serially on a single core. The value varies when using multiple processes, but the effects of the variation on the speedup estimate are small. Figure 1 shows the runtimes of the time-serial and the hybrid Parareal solution for different numbers of subdomains and corresponding MPI-processes. While the time-serial solution assigns each process to one core, the time-parallel solution assigns each process to one node and uses the eight cores inside the node for the temporal parallelization. The speedups actually achieved by the implementation are between 78% and 93% of this theoretical maximum, despite the overhead caused by the funneled mode, supporting the efficiency of the hybrid space-time parallel approach.

3.1.2 Total scaling.

As discussed above, one essential motivation for time-parallel schemes is to provide an additional direction of parallelization to achieve further reduction of time-to-solution after spatial parallelization saturates. Figure 2 shows the total speedup, that is compared against the time-serial solution on one core, for the time-serial and hybrid Parareal scheme. Because the considered problem is quite small and the

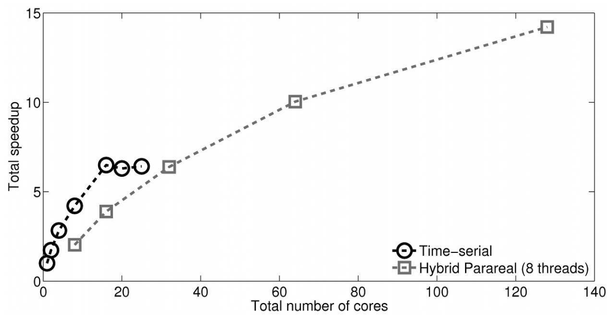

Fig. 2 Speedup achieved by the space-parallel, time-serial and the hybrid space-time-parallel scheme depending on the total number of used cores.

underlying stencil-based discretization is comparably cheap to evaluate in terms of computation time, the pure spatial parallelization scales only to 16 cores. Beyond that point, using more cores does not further reduce runtime. Also, near perfect scaling is seen only up to two cores, after this the parallel efficiency is noticeable less than one.

Note that the slow increase in speedup for the hybrid scheme is caused by the efficiency bound (3) of Parareal: For lower numbers of cores where the spatial parallelization is not yet saturated, the time-serial version performs better, because the efficiency of the parallelization in space, although no longer optimal, is still better than that of the time-parallel scheme. The advantage of the space-time-parallel scheme is that it can provide a significantly greater overall speedup. Hence, for a time-critical application where minimizing time-to-solution is of paramount importance and a purely spatial parallelization does not provide sufficient runtime reduction, a space-time parallel scheme can reduce runtime below some critical threshold if sufficient computational resources are available. The example clearly demonstrates the potential of the hybrid space-time parallelization to provide runtime reductions beyond the saturation of the space parallelization.

4 Conclusions

A shared memory implementation of the Parareal parallel-in-time integration scheme is combined with a standard distributed memory parallelization of a stencil-based spatial discretization. Parareal combines multiple time-steps into a coarser time-slice and iterates concurrent integration on multiple time-slices with a fast sequential correction. In the resulting hybrid space-time parallel scheme, each spatial subdomain is handled by one MPI-process which is assigned to one compute node. The time-slices from Parareal are assigned to different threads spawned by the process, with each thread running on one core of the node. Most prior works on Parareal focussed on the time-parallel aspect and there are only few papers on combinations of Parareal with space parallelization. In particular, the hybrid approach discussed here has not been employed before.

The capability of the hybrid scheme to provide runtime reduction beyond the saturation of the spatial parallelization is documented, supporting one of the main motivations for the study of time-parallelism, namely providing an additional direction of parallelization to minimize runtimes in time-critical applications. Also, the shared memory implementation of Parareal used here avoids the communication of volume data. A comparison with a pure MPI implementation would be an interesting point for future research.

Acknowledgments.

This work is funded by the Swiss "High Performance and High Productivity Computing" initiative HP2C.

References

1. Aubanel, E.: Scheduling of tasks in the parareal algorithm. Parallel Computing **37**, 172–182 (2011)
2. Bal, G., Maday, Y.: A "parareal" time discretization for non-linear PDE's with application to the pricing of an american put. In: L. Pavarino, A. Toselli (eds.) Recent Developments in Domain Decomp. Meth., *LNCSE*, vol. 23, pp. 189–202. Springer Berlin (2002)
3. Croce, R., Ruprecht, D., Krause, R.: Parallel-in-space-and-time simulation of the three-dimensional, unsteady Navier-Stokes equations for incompressible flow. Submitted to: Proc. of 5th Int. Conf. on High Performance Sci. Comp., Hanoi (2012). URL http://icsweb.inf.unisi.ch/preprints/preprints/file201203.pdf
4. Emmett, M., Minion, M.L.: Toward an efficient parallel in time method for partial differential equations. Comm. App. Math. and Comp. Sci. **7**, 105–132 (2012)
5. Gander, M.J., Vandewalle, S.: Analysis of the parareal time-parallel time-integration method. SIAM J. Sci. Comp. **29**(2), 556–578 (2007)
6. Lions, J.L., Maday, Y., Turinici, G.: A "parareal" in time discretization of PDE's. C. R. Acad. Sci. – Ser. I – Math. **332**, 661–668 (2001)
7. Maday, Y., Turinici, G.: The parareal in time iterative solver: A further direction to parallel implementation. In: R. Kornhuber, et al. (eds.) Domain Decomposition Methods in Science and Engineering, *LNCSE*, vol. 40, pp. 441–448. Springer, Berlin (2005)
8. Minion, M.L.: A hybrid parareal spectral deferred corrections method. Comm. App. Math. and Comp. Sci. **5**(2), 265–301 (2010)
9. Rabenseifner, R., Hager, G., Jost, G.: Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In: 17th Euromicro International Conference on Parallel, Distributed and Network-based processing, pp. 427–436 (2009)
10. Ruprecht, D., Krause, R.: Explicit parallel-in-time integration of a linear acoustic-advection system. Computers and Fluids **59**, 72–83 (2012)
11. Trindade, J.M.F., Pereira, J.C.F.: Parallel-in-time simulation of the unsteady Navier-Stokes equations for incompressible flow. Int J. Numer. Meth. Fluids **45**, 1123–1136 (2004)
12. Trindade, J.M.F., Pereira, J.C.F.: Parallel-in-time simulation of two-dimensional, unsteady, incompressible laminar flows. Num. Heat Trans., Part B **50**, 25–40 (2006)