Efficient implementation of a multi-level parallel in time algorithm

Matthew Emmett and Michael L. Minion

Abstract A strategy for scheduling the communication between processors in a multi-level parallel-in-time algorithm to reduce blocking communication is presented. The particular time-parallel method examined is the parallel full approximation scheme in space and time (PFASST), which utilizes a hierarchy of spatial and temporal discretization levels. By decomposing the update to initial conditions passed between processors into multiple spatial resolutions, the communication at the finest level can be scheduled to overlap with computation at coarser levels. The potential cost savings is demonstrated with a three dimensional PDE example.

1 Introduction

The last decade has seen an increase in research into the parallelization of numerical methods for ordinary and partial differential equations in the temporal direction. One strategy for temporal parallelization involves decomposing the solution into time slices, which are distributed across processors or groups of processors, and employing an iterative scheme for computing the solution on all time slices in parallel [4, 3, 2]. The communication between time slices in these algorithms is quite regular, where each processor must send updates to the initial condition to the processor representing the following time slice. This communication must be done during each iteration of the method, and the amount of data sent is proportional to the size of the problem being solved. Although this communication takes place less frequently than that which typically occurs in spatially parallelized solvers for PDEs, the size of the data that must be transmitted is relatively large, and hence, reducing the effective cost of this data transfer is necessary to avoid reduced parallel efficiency.

In [2] a new approach for the temporal parallelization of the numerical solution to partial differential equations, called the Parallel Full Approximation Scheme in Space and Time (PFASST), is introduced. PFASST is similar in structure to the earlier Parareal [4] and PITA [3] methods, but uses a deferred correction type procedure first described in [5, 7] within time slices instead of a traditional direct method, which provides an improved theoretical maximum parallel efficiency as compared

Matthew Emmett
Lawrence Berkeley National Laboratory, e-mail: `mwemmett@lbl.gov`

Michael L. Minion
Institute for Computational and Mathematical Engineering, Stanford University, e-mail: `mlminion@gmail.com`

to Parareal or PITA. The PFASST algorithm also uses a hierarchy of spatial and temporal discretizations of the problem, wherein coarse problems are defined in a procedure analogous to the full approximation scheme (FAS) used extensively in multigrid methods for nonlinear problems (see e.g., [1]). Since FAS is naturally recursive, an extension of the approach in [2] to multiple levels of spatial and temporal refinement is possible. The key algorithmic change in PFASST presented here concerns the issue of the communication cost.

The PFASST method is reviewed here in Sect. 2. In Section 3, an approach is outlined wherein corrections computed at different refinement levels are passed between processors in a way which can greatly reduce the communication overhead of the PFASST iterations. The timing results presented in Sect. 4 demonstrate the effectiveness of the proposed communication strategy. Finally, a short discussion of the current results and future research directions can be found in Sect. 5.

2 PFASST

In this section, a brief description of the PFASST algorithm is included. It is assumed that the reader is familiar with Spectral Deferred Correction (SDC) methods and full approximation scheme (FAS) corrections. For more complete details, see [7, 2].

For the following description, consider the ODE initial value problem

$$u'(t) = f(t, u(t)), \qquad u(0) = u_0, \tag{1}$$

where $t \in [0,T]$; $u_0, u(t) \in \mathbb{C}^N$; and $f : \mathbb{R} \times \mathbb{C}^N \to \mathbb{C}^N$. It is assumed here that (1) represents a method of lines discretization of a PDE.

For a PFASST computation with L levels of spatial and temporal resolution (with level 0 being the finest), the time interval of interest $[0,T]$ is divided into N uniform intervals $[t_n, t_{n+1}]$ which are assigned to the processors $\boldsymbol{P}_n$ where $n = 0 \dots N-1$. Each interval is subdivided on each level ℓ by defining $M_\ell + 1$ SDC nodes $\boldsymbol{t}_\ell = [t_{\ell,0} \cdots t_{\ell,M_\ell}]$ such that $t_n = t_{\ell,0} < \cdots < t_{\ell,M_\ell} = t_{n+1}$, where we have omitted the dependence of $\boldsymbol{t}_\ell$ on n for brevity. The SDC nodes $\boldsymbol{t}_{\ell+1}$ on level $\ell+1$ are chosen to be a subset of the SDC nodes $\boldsymbol{t}_\ell$ on level ℓ to facilitate interpolation and restriction between coarse and fine levels. Note that the use of point injection as the coarsening procedure with Gaussian quadrature nodes means that the coarse nodes may not correspond to Gaussian nodes. The solution at the m^{th} node on level ℓ during iteration k is denoted $\boldsymbol{U}(\ell,k,m)$. For brevity let $\boldsymbol{U}(\ell,k) = [\boldsymbol{U}(\ell,k,0), \cdots, \boldsymbol{U}(\ell,k,M_\ell)]$ and $\boldsymbol{F}(\ell,k) = [\boldsymbol{F}(\ell,k,0), \cdots, \boldsymbol{F}(\ell,k,M_\ell)] = [f(t_{\ell,0}, \boldsymbol{U}(\ell,k,0)), \cdots, f(t_{\ell,M_\ell}, \boldsymbol{U}(\ell,k,M_\ell))]$.

In the parareal method, the processors are typically initialized by using the coarse propagator in serial to yield a low-accuracy initial condition for each processor. In [2], an alternative initialization scheme is described. During initialization, each processor begins coarse SDC sweeps immediately using the initial condition from the first processor. Hence the number of coarse iterations (SDC sweeps) done on processor $\boldsymbol{P}_n$ in the initialization is equal to n rather than 1. This has the same total computational cost of doing one coarse SDC sweep per processor in serial, but the additional SDC sweeps can improve the accuracy of the solution significantly, as is demonstrated in [2]. During this initial iteration, no communication is necessary since each processor computes the same data as the processor corresponding to the previous process. Hence further discussion of the initialization procedure is omitted.

The full PFASST iterations for $k = 0 \dots K-1$ on each processor $\boldsymbol{P}_n$ proceed as follows. Assuming that the fine solution and function values $\boldsymbol{U}(0,k)$ and $\boldsymbol{F}(0,k)$ are available, the iterations are comprised of the following steps:

1. Perform one fine SDC sweep using the values $\boldsymbol{U}(0,k)$ and $\boldsymbol{F}(0,k)$. This will yield provisional updated values $\boldsymbol{U}(0,k+1)$ and $\boldsymbol{F}(0,k+1)$.
2. Send $\boldsymbol{U}(0,k+1,M_0)$ to processor $\boldsymbol{P}_{n+1}$ if $n < N-1$. This will be received as the new initial condition $\boldsymbol{U}(0,k+1,0)$ in the next iteration.
3. Go down the V-cycle: for each $\ell = 1 \dots L-2$
   a. Restrict the fine values $\boldsymbol{U}(\ell-1,k+1)$ to the coarse values $\boldsymbol{U}(\ell,k)$ and compute $\boldsymbol{F}(\ell,k)$.
   b. Compute the FAS correction $\boldsymbol{B}(\ell,k)$ using $\boldsymbol{F}(\ell-1,k+1)$, $\boldsymbol{F}(\ell,k)$, and $\boldsymbol{B}(\ell-1,k)$.
   c. Perform n_ℓ SDC sweeps with the values $\boldsymbol{U}(\ell,k)$, $\boldsymbol{F}(\ell,k)$ and the FAS correction $\boldsymbol{B}(\ell,k)$. This will yield new values $\boldsymbol{U}(\ell,k+1)$ and $\boldsymbol{F}(\ell,k+1)$.
   d. Send $\boldsymbol{U}(\ell,k+1,M_\ell)$ to processor $\boldsymbol{P}_{n+1}$ if $n < N-1$. This will be received as the new initial condition $\boldsymbol{U}(\ell,k+1,0)$ in the next iteration.
4. Perform the bottom sweep:
   a. Restrict the fine values $\boldsymbol{U}(L-2,k+1)$ to the coarse values $\boldsymbol{U}(L-1,k)$ and compute $\boldsymbol{F}(L-1,k)$.
   b. Compute the FAS correction $\boldsymbol{B}(L-1,k)$ using $\boldsymbol{F}(L-2,k+1)$, $\boldsymbol{F}(L-1,k)$, and $\boldsymbol{B}(L-2,k)$.
   c. Receive the new initial value $\boldsymbol{U}(L-1,k,0)$ from processor $\boldsymbol{P}_{n-1}$ if $n > 0$ and compute $\boldsymbol{F}(L-1,k,0)$.
   d. Perform n_{L-1} coarse SDC sweeps using the values $\boldsymbol{U}(L-1,k)$, $\boldsymbol{F}(L-1,k)$ and the FAS correction $\boldsymbol{B}(L-1,k)$. This will yield new values $\boldsymbol{U}(L-1,k+1)$ and $\boldsymbol{F}(L-1,k+1)$.
   e. Send $\boldsymbol{U}(L-1,k+1,M_{L-1)}$ to processor $\boldsymbol{P}_{n+1}$ if $n < N-1$. This will be received as the new initial condition $\boldsymbol{U}(L-1,k,0)$ in the current iteration on the next processor $\boldsymbol{P}_{n+1}$.
5. Return up the V-cycle: for each $\ell = L-2 \dots 1$:
   a. Interpolate coarse correction $\boldsymbol{U}(\ell+1,k+1) - \boldsymbol{U}(\ell+1,k)$ in space and time and add to $\boldsymbol{U}(\ell,k+1)$. Recompute new values $\boldsymbol{F}(\ell,k+1)$.
   b. Receive the new initial value $\boldsymbol{U}(\ell,k+1,0)$ from processor $\boldsymbol{P}_{n-1}$ if $n > 0$.
   c. Interpolate correction $\boldsymbol{U}(\ell+1,k+1,0) - \boldsymbol{U}(\ell+1,k,0)$ to new $\boldsymbol{U}(\ell,k+1,0)$ and recompute $\boldsymbol{F}(\ell,k+1,0)$.

   d. Perform n_ℓ SDC sweeps with the values $\boldsymbol{U}(\ell,k+1)$, $\boldsymbol{F}(\ell,k+1)$ and the FAS correction $\boldsymbol{B}(\ell,k)$. This will once again yield new values $\boldsymbol{U}(\ell,k+1)$ and $\boldsymbol{F}(\ell,k+1)$.

6. Interpolate coarse correction $\boldsymbol{U}(1,k+1)-\boldsymbol{U}(1,k)$ in space and time and add to $\boldsymbol{U}(0,k+1)$. Recompute new values $\boldsymbol{F}(0,k+1)$.
7. Receive the new initial value $\boldsymbol{U}(0,k+1,0)$ from processor $\boldsymbol{P}_{n-1}$ if $n>0$.
8. Interpolate correction $\boldsymbol{U}(1,k+1,0)-\boldsymbol{U}(1,k,0)$ to new $\boldsymbol{U}(0,k+1,0)$ and recompute $\boldsymbol{F}(0,k+1,0)$.

The steps above are illustrated in Figure 1(b), in which solid blocks denote SDC sweeps (F_ℓ) and gradient blocks denote interpolation ($I_{\ell+1}^{\ell}$) or restriction ($R_{\ell}^{\ell+1}$). The length of the blocks are proportional to their cost, with fine SDC sweeps being 4 and 16 times more expensive than intermediate and coarse SDC sweeps, respectively (which would correspond to a 1D PFASST scheme with both spatial and temporal refinements by a factor of 2). The length of the interpolation and restriction blocks is also proportional to their cost: when transferring between levels we must re-evaluate the function values $\boldsymbol{F}(\ell,k)$ in order to compute the FAS corrections $\boldsymbol{B}(\ell,k)$.

3 Communication between processors

In the precursors to this work appearing in [7, 2] as well as the original papers on the parareal method, little attention is given to the topic of scheduling the communication between processors. In this section, a strategy which effectively unblocks communication except at the coarsest resolution is presented. It is assumed here that the parallel implementation of PFASST allows computation and communication to be performed simultaneously.

In each PFASST iteration, the full solution (or correction to the solution) must be passed forward in time to the next processor. In the two level scheme presented in [7, 2], this is done directly after the coarse correction is applied to the current fine solution. Since the coarse SDC sweep on the next processor cannot begin until this data is received, scheduling the communication in this way results in a *blocking* communication. The blocking communication is depicted in Fig. 1(a), where each column represents the operations done on a processor with wall time progressing from bottom to top. The white and black circles correspond to the send and receive process on each processor. After the coarsest SDC sweep (denoted $\mathscr{F}_1$), the full update of the initial condition is sent forward in time. The white gap represents the waiting time, which grows linearly with the number of processors.

Note in Fig. 1(a), the first operation performed after a processor receives data is a coarse SDC sweep. In order to perform this sweep, a new initial condition is required, but only at the coarse resolution. The key observation used here is that it is only necessary to pass the corrections to the initial data during each PFASST iteration, and more importantly this communication can be decomposed into corrections corresponding to each level of spatial resolution. Although this means that more data in total is being passed during each PFASST iteration, data from the finer levels can be sent before the corresponding fine SDC sweeps are performed on each processor. Therefore, if the computational cost of the computation at the coarser levels is greater than the communication cost of sending data at a particular level, then the communication becomes *non-blocking*.

For example, consider Fig. 1(b), which diagrams the scheduling of communication for a three-level implementation of the PFASST algorithm. At each level, as soon as an SDC sweep is completed (denoted by $\mathscr{F}_\ell$ for $\ell = 0 \dots 2$), the correction to the solution at the final SDC node (which corresponds to the first SDC node on the next processor) is sent. This can be done before the recursive call to compute a correction at the next coarsest level (denoted by the blocks $R_\ell^{\ell+1}$). The sent data can then be received in a buffer at the next processor and is not needed until after the corresponding coarse correction has been computed (denoted by the blocks $I_{\ell+1}^{\ell}$) on that processor. Hence, the sending of the finest data overlaps with the computation of the correction on two coarser levels. It is only at the coarsest level that there is no computational work to be done while waiting for the data to be received. However, if the coarse data is significantly smaller than fine data, the communication cost at the coarsest level is likewise significantly reduced. In the three-level, three-dimensional example in Sect. 4, the coarsest level contains 1/64 the amount of data as the finest level with communication time similarly reduced.

It should be noted that the crossing of the lines corresponding to communication in Fig. 1(b) assume that blocking coarse communication could be scheduled to interrupt non-blocking fine level communication, a feature which may not exist in a standard message passing library. If this is not the case, there is still the opportunity to overlap computation with communication before the blocking coarsest level send occurs. Finally, recall that the work performed by each processor in Fig. 1(b) is not uniform since processor $\boldsymbol{P}_n$ does n coarse SDC sweeps during the initialization procedure.

4 Timing

Timing information for a three-level PFASST run was obtained for a three dimensional model problem: the incompressible Navier-Stokes equations given by

$$\boldsymbol{u}_t + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = \nu \nabla^2 \boldsymbol{u} - \nabla p \nabla \cdot \boldsymbol{u} = 0. \tag{2}$$

A method of lines approach is employed by placing the equations in projection form and using spectral approximations to all spatial derivatives via the FFT [6]. The advective piece of the equation is treated explicitly while the diffusive piece is treated implicitly. The fine spatial discretization consists of 256^3 points in a unit cube, resulting in a total of 3×256^3 degrees of freedom on the fine level, or approximately 384 megabytes using 64 bits per degree of freedom. The fine temporal discretization consists of 5 Gauss-Lobatto SDC nodes. The run was performed across 16 proces-

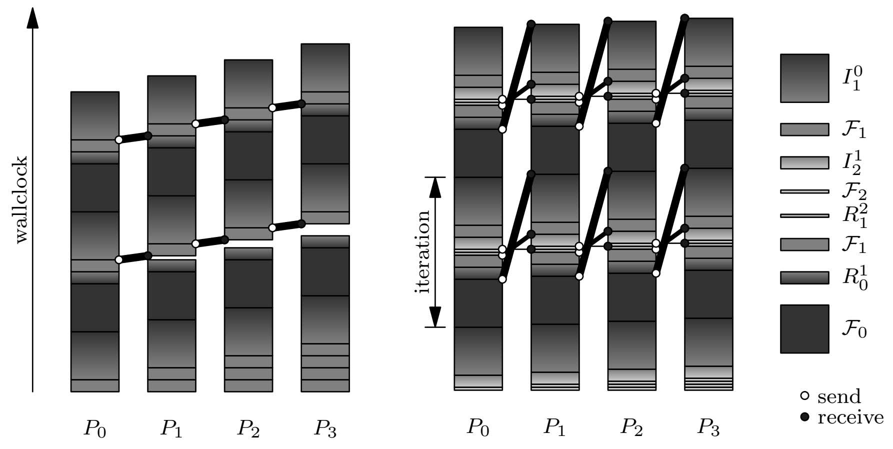

Fig. 1 Left: (a) Communication diagram for the original 2-level PFASST algorithm. Right: (b) Communication diagram for the 3-level V-cycle PFASST algorithm.

sors of "Edison", the Cray XC30 system at the National Energy Research Scientific Computing Center (NERSC).

Figs. 2 and 3 present timing information for various parts of the PFASST algorithm across the processors for each PFASST iteration. From Fig. 2 we note that the iteration time (which encompasses all overhead costs including interpolation, restriction, and FAS computation) is fairly consistent across each processor and iteration, and that the cost of the intermediate and coarse sweeps are significantly cheaper than the fine sweep.

From Fig. 3 we note that the (blocking) coarse send and receive times are fairly significant (send/receive 0) between some processors. This establishes that communication across compute nodes is non-trivial even at the coarse level (recall that the coarse level consists of 3×64^3 degrees of freedom, which is 64 times less than the fine level). Finally, the fine and intermediate send and receive times (send/receive 1 and 2) are essentially zero across all processors and iterations. This demonstrates that the fine and intermediate communications are essentially non-blocking and were successfully overlapped with computation.

The three-level PFASST run using 16 time processors desribed above achieves a speedup of roughly 7.2 compared to a serial SDC-based run (which requires 8 serial iterations per time step to acheive the same accuracy as 6 PFASST iterations). This corresponds to a parallel efficiency of roughly 45%. The parallel efficiency of PFASST can vary substantially depending on the number of processors, the error tolerance, and the sensitivity of the problem at hand [2].

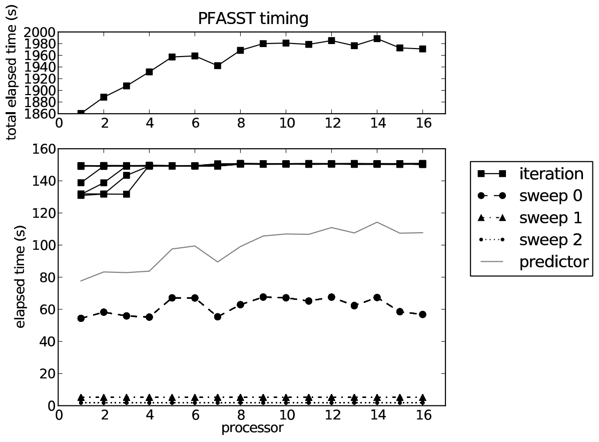

Fig. 2 Timing information for the three-dimensional Navier-Stokes solver. The top panel shows the total elapsed run time. The bottom panel shows iteration time (including all overhead), SDC sweep time for each level, and the initialization time (predictor).

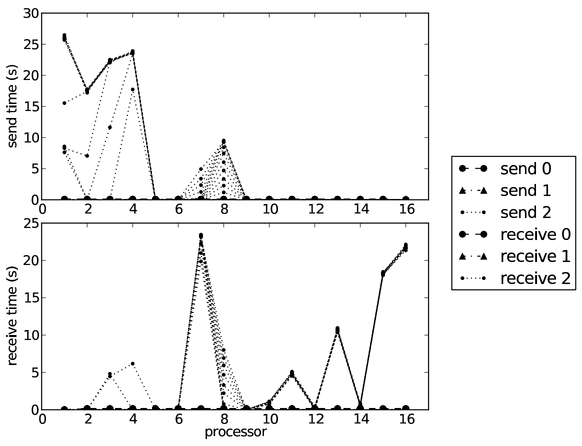

Fig. 3 Communication timing information for the three-dimensional Navier-Stokes solver. The top panel shows send time for each level, and the bottom panel shows receive time for each level. Note that the send and receive times for the intermediate and fine levels (1 and 0) are negligible compared to the coarse level (2).

5 Discussion

In summary, we have demonstrated how the necessary transfer of relatively large amounts of data between processors in the PFASST algorithm can be scheduled so that only a small amount of the transfer is blocking. As long as the computation involved in a recursive call to a coarser level correction is more expensive than the communication, the communication cost is negligible. The effectiveness of the scheduling procedure relies on the communication and computation being done simultaneously, and is optimal if blocking communication can interrupt non-blocking communication between two processors.

Current trends in the design of the next generation of large parallel computers suggest that the relative cost of data transfer between processors will continue to grow. In this case, more elaborate strategies to avoid blocking communication in the PFASST algorithm might become necessary. For example, since only the correction to the solution needs to be passed between processors, it is possible that fewer significant digits could be used to transmit data. The main point we stress here is that, except at the coarsest level, there is useful work that a processor can perform while data is being passed from processor to processor. In fact, the algorithm could be reconfigured so that at each stage of the FAS procedure, SDC sweeps are performed at each level until the necessary data at the next finest level is received.

References

1. Briggs, W.L., Henson, V.E., McCormick, S.F.: A Multigrid Tutorial, vol. 72. SIAM (2000)
2. Emmett, M., Minion, M.: Toward an efficient parallel in time method for partial differential equations. Communications in Applied Mathematics and Computational Science **7**(1), 105–132 (2012)
3. Farhat, C., Chandesris, M.: Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications. Internat. J. Numer. Methods Engrg. **58**(9), 1397–1434 (2003)
4. Lions, J., Maday, Y., Turinici, G.: A ”parareal” in time discretization of PDE’s. Comptes Rendus de l’Academie des Sciences Series I Mathematics **332**(7), 661–668 (2001)
5. Minion, M., Williams, S.: Parareal and spectral deferred corrections. In: AIP Conference Proceedings, vol. 1048, pp. 388–391. AIP (2008)
6. Minion, M.L.: Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. Appl. Numer. Math. **48**(3-4), 369–387 (2004)
7. Minion, M.L.: A hybrid parareal spectral deferred corrections method. Comm. Appl. Math. and Comp. Sci. **5**(2), 265–301 (2010)