RIDC-DD: A parallel space-time algorithm

Ronald Haynes [α] and Benjamin Ong [β]
[α] Memorial University of Newfoundland, Saint Johns, NL, Canada
[β] Michigan State University, East Lansing, MI, USA

We present recent efforts to develop fully parallel space–time algorithms, which will improve the fidelity and scalability of numerical simulations across many areas in computational science. Our approach combines Revisionist Integral Defect Correction (RIDC) methods [2, 3], a family of time-parallel integrators, with Domain Decomposition (DD) methods [6, 5], an approach to split boundary value problems into smaller boundary value problems on sub-domains and iterating to coordinate the solution between adjacent sub-domains. The basic idea is as follows: while a set of N processing cores are computing a low-order approximation to the domain-decomposed solution at time t^n, additional sets of N cores are simultaneously computing corrections to the low(er)-order approximations at time t^m, where $m < n$.

Consider the general problem of interest:

$$u_t = \mathcal{L}(t,u), \quad x \in \Omega \times [0,T] \tag{1a}$$
$$\mathcal{B}(t,u) = 0, \quad x \in \partial\Omega \times [0,T] \tag{1b}$$
$$u(0,x) = g(x), \quad x \in \Omega. \tag{1c}$$

Denote an approximate solution to (1) as $\eta(t,x)$, and the residual as $\epsilon(t,x) = \eta_t - \mathcal{L}(t,\eta)$. Then, the following correction PDE can be derived,

$$\left[e + \int_0^t \epsilon(\tau,x)\,d\tau\right]_t = \mathcal{L}(t,\eta+e) - \mathcal{L}(t,\eta), \quad x \in \Omega \times [0,T] \tag{2a}$$
$$e(t,x) = 0, \quad x \in \partial\Omega \times [0,T] \tag{2b}$$
$$e(0,x) = 0, \quad x \in \Omega, \tag{2c}$$

where e is the correction to an approximate solution η. *Pipeline* parallelism can be employed by staggering solutions to the PDE of interest (1) and multiple correction PDEs (2) (after initial start-up costs).

Data parallelism is layered on top of the time (pipeline) parallelism through DD. As a concrete example, discretize (1) and (2) using a first-order backwards Euler integrator. After algebraic manipulations, one recovers a model boundary value problem from both (1) and (2),

$$(1 - \alpha\mathcal{L})u = f(x), \quad x \in \Omega \tag{3a}$$
$$\mathcal{C}(u) = 0, \quad x \in \partial\Omega. \tag{3b}$$

Decomposing the computational domain Ω into N non-overlapping domains, $\Omega = \bigcup_{i=1}^{N} \Omega_i$, one reformulates (3) into the coupled system of boundary value problems. Also known as a Schwarz-type approach, the reformulated problem is to find $(u_i)_{1\le i\le N}$ such that

$$(1 - \alpha\mathcal{L})u_i = f(x), \quad x \in \Omega_i \tag{4a}$$
$$\mathcal{C}(u_i) = 0, \quad x \in \partial\Omega_i \cap \partial\Omega, \tag{4b}$$
$$\mathcal{T}_{ij}(u_i) = \mathcal{T}_{ij}(u_j), \quad x \in \partial\Omega_i \cap \partial\Omega_j, \tag{4c}$$

where $(\mathcal{T}_{ij})_{1\le i,j\le N}$ are transmission conditions on the interfaces between the sub-domains. The coupled system (5) is solved iteratively using a Jacobi algorithm,

$$(1 - \alpha\mathcal{L})u_i^{k+1} = f(x), \quad x \in \Omega_i \tag{5a}$$
$$\mathcal{C}(u_i^{k+1}) = 0, \quad x \in \partial\Omega_i \cap \partial\Omega, \tag{5b}$$
$$\mathcal{T}_{ij}(u_i^{k+1}) = \mathcal{T}_{ij}(u_j^{k}), \quad x \in \partial\Omega_i \cap \partial\Omega_j. \tag{5c}$$

Since this formulation of a parallel space–time algorithm involves pipeline and data parallelism, our implementation uses multiple modes of programming. OpenMP is used to handle the pipeline (time) parallelism; MPI is used to handle the data (DD) parallelism. The hybrid OpenMPI–MPI framework used to is discussed in [1, 4]. The C++ software was compiled and bench-marked on a general purpose x86 cluster. Each node, consisting of dual-socket Intel Sandy Bridge processors, are connected using FDR infiniband. The linear heat equation in 2D with optimized transmission conditions was used as a test problem.

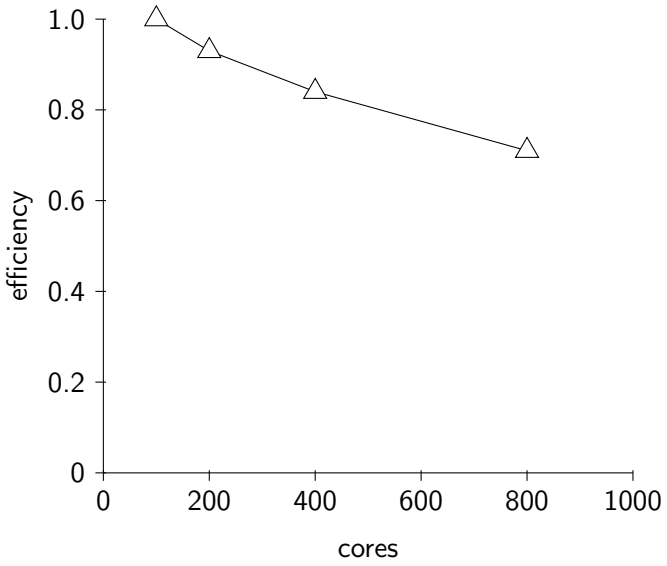

Figure 1: The linear heat equation is solved in $\mathbb{R}^2$. The domain is subdivided into a 10×10 grid of non-overlapping subdomains. A second order finite discretization is used spatially, and an eighth-order RIDC method is used for the time discretization. The number of concurrent threads that the RIDC integrator is allowed to use is varied. This result shows that we can take a spatially parallel code, and layer on time-parallelism with relatively high efficiency.

References

[1] A. Christlieb, R. Haynes, and B. Ong. A parallel space-time algorithm. *SIAM J. Sci. Comput.*, 34(5):233–248, 2012.

[2] A. Christlieb, C. Macdonald, and B. Ong. Parallel high-order integrators. *SIAM J. Sci. Comput.*, 32(2):818–835, 2010.

[3] A. Christlieb and B. Ong. Implicit parallel time integrators. *J. Sci. Comput.*, 49(2):167–179, 2011.

[4] R. Haynes and B. Ong. A hybrid MPI-OpenMP algorithm for the parallel space-time solution of time dependent PDEs. In *Domain Decomposition Methods in Science and Engineering XXI*, 2013. to appear.

[5] T. Mathew. *Domain decomposition methods for the numerical solution of partial differential equations*, volume 61 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 2008.

[6] A. Toselli and O. Widlund. *Domain decomposition methods—algorithms and theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2005.