Leveraging the Task-Aware GASPI's One-Sided Communications in Saiph

Sandra Macià*, Kevin Sala† and Vicenç Beltran‡
Barcelona Supercomputing Center (BSC)
Barcelona, Spain
Email: *sandra.macia@bsc.es, †kevin.sala@bsc.es, ‡vbeltran@bsc.es

***Abstract*—Saiph is a Domain-Specific Language (DSL) that eases the simulation of physical phenomena from the Computational Fluid Dynamics (CFD) domain in HPC environments. Saiph offers a high level of abstraction and a numerical library implementing parallel Finite Difference Methods (FDM) and explicit time solvers. Internally, Saiph features multiple back-ends to generate different parallel versions of high-level applications. There is one back-end implemented only with MPI and another back-end implemented with MPI and OmpSs-2 tasks that leverages the Task-Aware MPI library. The MPI-only approach manually overlaps computations and communications using non-blocking MPI primitives. In contrast, the hybrid TAMPI approach fully taskifies both computations and communications using task data dependencies and the TAMPI library, allowing the natural and automatic overlap of both phases.**

In this technical report, we extend Saiph with a new back-end that generates a hybrid task-based GASPI+OmpSs-2 variant that leverages the one-sided communications provided by the Task-Aware GASPI library. The one-sided (or RMA) communications allow the user applications to reduce the communication granularity and scale better to a high number of computing nodes.

I. Saiph

Saiph [1] [2] is a Domain-Specific Language (DSL) that eases the simulation of physical phenomena from the Computational Fluid Dynamics (CFD) domain in HPC environments. The tool provides a high level of abstraction and a numerical library implementing parallel Finite Difference Methods (FDM) and explicit time solvers. Saiph automatically discretizes a spatial domain mapping continuous field information into a Cartesian grid of points. The temporal dimension is discretized through a time-stepping loop at which each iteration defines the state of a system using previous time-step states. Computationally, a time-step loop encloses a spatial traversal at which linear algebra and stencil computations occur. Moreover, the spatial loop is automatically blocked to ensure good memory locality while providing large amounts of potential parallel work. Since calculations befall over neighboring mesh point values from previous time-iterations: (i) the absence of data dependencies within time-steps ensures the spatial loop's embarrassingly parallelism and (ii) the distributed executions using a domain decomposition approach involve boundaries communications between neighboring processes at the end of each time-step.

Saiph features multiple back-ends so that it can generate different parallel versions from the same high-level application. The current back-ends are an optimized pure MPI [3] approach and a hybrid task-based MPI+OmpSs-2 that leverages the Task-Aware MPI library [4] [5]. The MPI-only approach manually overlaps computations and communications using non-blocking MPI primitives. In contrast, the hybrid TAMPI approach taskifies with OmpSs-2 [6] both computations and communications using task data dependencies and the TAMPI library. This full taskification follows a data-flow execution model and allows the natural and automatic overlap of both phases.

II. Porting Saiph to the Task-Aware GASPI

We extend Saiph and develop a new back-end based on the same taskification as the TAMPI+OmpSs-2 version but using Task-Aware GASPI (TAGASPI) [7] [8] for communications. The TAGASPI library works on top of the GASPI model [9], which offers a simple API to write/read to/from other processes' memory directly without the intervention of the target process. TAGASPI integrates these fine-grained RMA operations with the OmpSs-2 tasking model, allowing tasks to safely and efficiently issuing RMA operations concurrently.

For the new TAGASPI back-end, we adapt computation tasks to include data unpack and pack operations. After any unpack, receivers use `tagaspi_notify` to send an ack notification to the sender side allowing the sender to initiate the next data writing. Consequently, writer tasks on the sender side asynchronously wait for the corresponding ack notification using the `onready` clause [7] before starting the remote write operation. The ack notification prevents the overwriting of the data sent at the previous iteration before being consumed by the receiver. Moreover, communication tasks involve sending and receiving, to and from, different neighbor processes making use of `tagaspi_write_notify` to write and notify the receiver side and using `tagaspi_notify_iwait` to wait asynchronously for the remote notification of the transfer.

References

[1] S. Macià, S. Mateo, P. J. Martínez-Ferrer, V. Beltran, D. Mira, and E. Ayguadé, "Saiph: Towards a dsl for high-performance computational fluid dynamics," in *Proceedings of the Real World Domain Specific Languages Workshop*, 2018, pp. 1–10.

[2] S. Macià, P. J. Martínez-Ferrer, S. Mateo, V. Beltran, and E. Ayguadé, "Assembling a high-productivity dsl for computational fluid dynamics," in *Proceedings of the Platform for Advanced Scientific Computing Conference*, 2019, pp. 1–11.

[3] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard. Version 3.1*. University of Tennessee, Jun. 2015.

[4] K. Sala, X. Teruel, J. M. Perez, A. J. Peña, V. Beltran, and J. Labarta, "Integrating blocking and non-blocking MPI primitives with task-based programming models," *Parallel Computing*, vol. 85, pp. 153–166, 2019.

[5] Barcelona Supercomputing Center, "Task-Aware MPI (TAMPI) Library." [Online]. Available: https://github.com/bsc-pm/tampi

[6] ——, "OmpSs-2 Specification," accessed: 2021-07-28. [Online]. Available: https://pm.bsc.es/ompss-2-docs/spec/

[7] K. Sala, S. Macià, and V. Beltran, "Combining one-sided communications with task-based programming models," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2021 (In Press).

[8] Barcelona Supercomputing Center, "Task-Aware GASPI (TAGASPI) Library." [Online]. Available: https://github.com/bsc-pm/tagaspi

[9] GASPI Forum, "GASPI: Global Address Space Programming Interface. Version 17.1," February 7th 2017, available at: http://www.gaspi.de/gaspi/. Accessed: 2021-07-28.