The most promising programming model for future supercomputers is expected to be based on asynchronous communication to hide latency and on a partitioned global address space. It has to support fault tolerance.

GPI provides such an API for your standard language. It delivers the full hardware speed directly to the application. The truly asynchronous, one sided and zero copy semantics of GPI allows to overlap communication and computation perfectly, a key requirement for scalable software.

GPI is easy to use and the adoption of existing parallel codes does not require a complete rewrite. In GPI the communication pattern is under full control. This is in contrast to PGAS languages like CoArray Fortran or UPC, where one often ends up with a major rewrite and still depends on compiler magic.

In order to achieve good scalability, one has to rethink and reformulate the communication strategy. The elimination of synchronous communication is beneficial not only for GPI programs, but often improves message passing implementations as well.

**GPI coverage of a large class of scientific algorithms**

- Computational fluid dynamics (1)
- Stencil codes (2)
- Graph algorithms (3)

**Best in class performance in all domains**

- Full overlap of communication and computation
- No CPU cycles spent for communication
- No cache disturbance
- Truly asynchronous
- Scalability
GPI enables strong scalability of the next generation CFD solver FLUCS far beyond the capabilities of MPI. FLUCS is developed by the German Aerospace Research Center (DLR). It focuses on massively parallel simulations of compressible flows on unstructured grids using hybrid parallelization based on a 2-level domain decomposition. It features a finite-volume as well as a discontinuous Galerkin (DG) method to solve the Euler/Navier-Stokes/RANS equations. GPI makes a truly asynchronous implementation possible that perfectly overlaps computation and communication. As such, running FLUCS on a coarse mesh consisting of less than 2 million elements, GPI allows using 200 cluster nodes with 80% parallel efficiency, i.e. merely 200 elements per thread. With hybrid MPI, using the same communication pattern, only 80 nodes can be used at the same efficiency. With flat MPI, this reduces to 40 cluster nodes which can be used at 80% efficiency. The GPI implementation clearly outperforms the MPI implementation.

**Stencil codes: Berlin Quantum Chromo Dynamics**

GPI is able to fully hide the communication behind the computation in the theoretically feasible regime as seen for the Berlin Quantum Chromo Dynamics code (BQCD). BQCD is a Hybrid Quantum Monte-Carlo algorithm using domain decomposition of the underlying four dimensional regular space-time grid. The computational kernel of BQCD is the evaluation of a four dimensional nearest neighbor stencil operator. In contrast to the MPI implementation, the GPI implementation shows perfect overlap of computation and communication in the theoretically feasible regime. No cache coherency effects and no CPU load is induced by the communication. There is essentially no difference between the theoretical expectation and the practical measurement.

**Graph algorithms: Unbalanced Tree Search**

GPI outperforms MPI in the Unbalanced Tree Search (UTS) benchmark by a maximum factor of 2.5 in terms of raw performance. The UTS benchmark is designed to evaluate the performance and the ease of programming for parallel applications requiring dynamic load balancing. These applications are characterized by unpredictable communication, unpredictable synchronization and dynamic work granularity. The one sided communication paradigm of GPI perfectly fits these requirements and allows for the fastest UTS implementation ever.

---