Abstract This paper presents a scalable and efficient Message-Passing in Java (MPJ) collective communication library for parallel computing on multi-core architectures. The continuous increase in the number of cores per processor underscores the need for scalable parallel solutions. Moreover, current system deployments are usually multi-core clusters, a hybrid shared/distributed memory architecture which increases the complexity of communication protocols. Here, Java represents an attractive choice for the development of communication middleware for these systems, as it provides built-in networking and multithreading support. As the gap between Java and compiled languages performance has been narrowing for the last years, Java is an emerging option for High Performance Computing (HPC).

Our MPJ collective communication library increases Java HPC applications performance on multi-core clusters: (1) providing multi-core aware collective primitives; (2) implementing several algorithms (up to six) per collective operation, whereas publicly available MPJ libraries are usually restricted to one algorithm; (3) analyzing the efficiency of thread-based collective operations; (4) selecting at runtime the most efficient algorithm depending on the specific multi-core system architecture, and the number of cores and message length involved in the collective operation; (5) supporting the automatic performance tuning of the collectives depending on the system and communication parameters; and (6) allowing its integration in any MPJ implementation as it is based on MPJ point-to-point primitives. A performance eval-
uation on an InfiniBand and Gigabit Ethernet multi-core cluster has shown that the implemented collectives significantly outperform the original ones, as well as higher speedups when analyzing the impact of their use on collective communications intensive Java HPC applications. Finally, the presented library has been successfully integrated in MPJ Express (http://mpj-express.org), and will be distributed with the next release.

**Keywords** Message-passing in Java (MPJ) · Multi-core clusters · Scalable collective communication · High performance computing · Performance evaluation

1 Introduction

Java is the leading programming language both in academia and industry environments, and it is an alternative for High Performance Computing (HPC) [1] due to its appealing characteristics: built-in networking and multithreading support, object orientation, automatic memory management, platform independence, portability, security, an extensive API and a wide community of developers, and besides it is the main training language for computer science students. Moreover, performance is no longer an obstacle. Java, in its early days, was severely criticized for its poor computational performance, reported to be within a factor of four of the equivalent Fortran code in [2]. However, currently, thanks to advances in JVMs and Just-In-Time (JIT) compilation, which are able to generate native executable code from the platform independent bytecode, Java performance is around a 30% slower on average than natively compiled languages (e.g., C and Fortran), according to [1] and [3]. This relatively low overhead trades off for the interesting features of Java. However, although this performance gap is relatively small, it can be particularly high for communication-intensive parallel applications when relying on poorly scalable Java communication libraries, which has hindered Java adoption for HPC. Thus, this paper presents a more scalable collectives communication library.

Message-passing is the most widely used parallel programming paradigm as it is highly portable, scalable and usually provides good performance. It is the preferred choice for parallel programming distributed memory systems such as multi-core clusters, currently the most popular system deployments due to their scalability, flexibility and interesting cost/performance ratio. Here, Java represents an attractive alternative to languages traditionally used in HPC, such as C or Fortran together with their MPI bindings, for the development of applications for these systems as it provides built-in networking and multithreading support, key features for taking full advantage of hybrid shared/distributed memory architectures. Thus, Java can use threads in shared memory (intra-node) and its networking support for distributed memory (inter-node) communications.

The increasing number of cores per system demands efficient and scalable message-passing communication middleware. However, up to now Message-Passing in Java (MPJ) implementations have been focused on providing production-quality implementations of the full MPJ specification, rather than concentrate on developing scalable collective communications. MPJ application developers use collective primitives for performing standard data movements (e.g., broadcast, scatter and gather)