1 Introduction

Future exascale computer systems will be capable of delivering $10^{18}$ operations per second (1 ExaFLOP) using the High Performance Linpack (HPL) benchmark. Despite the precise system architectures of such supercomputers are still not clear, one can safely assume that they will consist of approximately $10^9$ computing units. One of the major challenges of programming exascale supercomputers is to handle effectively an unprecedented amount of concurrency. In particular, collective and synchronization operations involving $10^9$ processes are a daunting task.

The Message Passing Interface (MPI) has emerged as the most common approach for parallel programming on current petascale machines. MPI specifies the message-passing parallel programming model, where data is passed from the local address space of one process to the address space of another process using explicit exchange of messages. However, it is unclear if MPI will perform efficiently on exascale machines. Also, Partitioned Global Address Space (PGAS) languages and libraries are increasingly being considered as alternatives or complements to MPI. In the PGAS programming model, the memory is divided logically in two partitions: one is global and accessible by all the processes, and the other one is local to the process. Communication in PGAS model is one-sided (it does not require explicit cooperation of the receiving process) and it is based on Remote Direct Memory Access.

2 Our Approach

In this poster, we present the initial work done in the EC-funded FP7 EPiGRAM project on exascale programming models (http://www.epigram-project.eu/). We focus on the challenges of Message Passing and PGAS programming models in the exascale era. The new features of MPI3.0 [1], such as sparse collectives and non-blocking collectives, are described and discussed. Missing features, such as fault tolerance, and current limitations of MPI (non scalability of the all-to-all communication operations) for exascale are considered. In addition, we present the main features of the PGAS GPI library [2], developed by Fraunhofer Society, and discuss its potential for exascale focusing on features such as asynchronous and one-sided communication and fault tolerance.

In summary, we present an investigation of the two most promising programming models for exascale, Message Passing and PGAS. The new features that might deliver exascale performance are discussed, and the current limitations of these two approaches are outlined.

References
