In the above code, there is only one iteration, it is used to demonstrate how to define range reference with *halo* area, and how to use the `writeHalo` function.

4 Project in progress

The related projects of our work may include development of MPI, HPF, and other parallel languages such as ZPL and Spar, which are introduced else where[6]. Here we explain more backgrounds and future developments about our own project.

The work originated in our compilation practice in HPF. As introduced in [2], our compiler emphasize runtime system support. *Adlib*[6], as PCRC runtime kernel library, provides a rich set of collective communication functions. During the practice, it is realized that the runtime interface can be effectively raised to a higher level, and a rather straight-forward (compared to HPF) compiler can be developed to translate the high level language code to a node program calling runtime interface functions.

Currently, Java interface has been implemented on top of the Adlib library. With classes such as `Group`, `Rang` and `Location` in the Java interface, one can write Java programs quite similar to HPJava we proposed here. Yet, the program executed in this way will have large overhead due to function calls (such as address translation) when accessing data inside loop constructs.

Given the knowledge of data distribution plus inquiry functions inside runtime library, one can substitute address translation calls with linear operation on the loop variable, and keep most of the inquiry function calls out side loop constructs. This is the basic idea of the HPJava compiler.

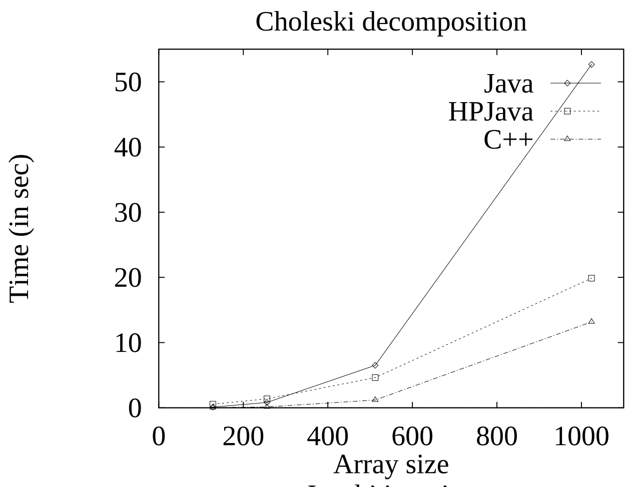
At present time, we are working on the design and implementation of the prototype of this kind "translator". Further research works may include optimization and safety-checking techniques in the compiler for HP-spmd programming.

Figure 3 shows a preliminary benchmark for hand translated codes of our examples. The parallel programs are executed on 4 sparc-sun-solaris2.5.1 with mpich MPI and Java JIT compiler in JDK 1.2Beta2. For Jacobi iteration, the timing is for about 90 iterations.

We also compared the sequential C++ version of the code. As shown in the figure.

Similar test was made on an 8-node SGI challenge(mips-sgi-irix6.2), the communication time is much smaller than the one on solaris, due to MPI device using shared memory. Yet the overall performance is not as good, because the JIT compiler is not supported on irix. The whole system are being ported to Windows NT, where we may use both shared memory and JIT techniques.

[6]For more analysis, please refer to our documents at http://www.npac.syr.edu/projects/pcrc/doc



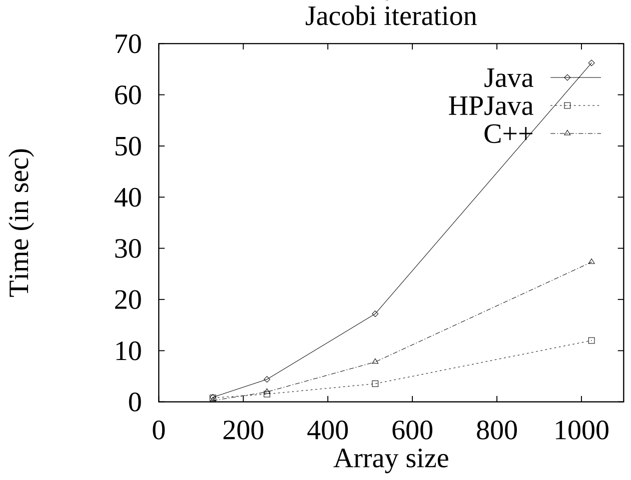


Figure 3: Preliminary performance

5 Summary

Through the simple examples in the report, we can see the programming language presented here has the flexibility of a SPMD program, and the convenience of HPF. The language encourages programmers to express parallel algorithms in a more explicit way. We believe it will help programmers to solve real application problems easier compared with using communication packages such as MPI directly, and allow the compiler writer to implement the language compiler without the difficulties met in the HPF compilation.

The Java binding is only an introduction of the new programming style. (A Fortran binding is being developed.) It can be used as a software tool for teaching parallel programming. And as Java for scientific computation become more mature, it will be a practical programming language to solve real application problems in parallel and distribute environments.

References

[1] High Performance Fortran Forum, "High Performance Fortran Language Specification", version 2.0, Oct. 1996

[2] Guansong Zhang, Bryan Carpenter, Geoffrey Fox, Xiaoming Li, Xinying Li, and Yuhong When. "PCRC-based HPF compilation", 10th International Workshop on Languages and Compilers for Parallel Computing, 1997.

[3] Bryan Carpenter, Guansong Zhang, Geoffrey Fox, Xinying Li, and Yuhong Wen. "Introduction to Java-Ad". http://www.npac.syr.edu/projects/pcrc/doc.

[4] R. Das, M. Uysal, J.H. Salz, and Y.-S. Hwang. "Communication optimizations for irregular scientific computations on distributed memory architectures". Journal of Parallel and Distributed Computing, Sep. 1994

[5] J. Nieplocha, R.J. Harrison, and R.J. Littlefield. The Global Array: Non-uniform memory access programming model for high-perfomance computers. The Journal of Supercomputing, 1996.

[6] Bryan Carpenter, Guansong Zhang and Yuhong Wen, "NPAC PCRC Runtime Kernel (Adlib) definition", http://www.npac.syr.edu/projects/pcrc/doc