XcalableMP

Directive-based Language eXtension for Scalable and Performance-aware Parallel Programming

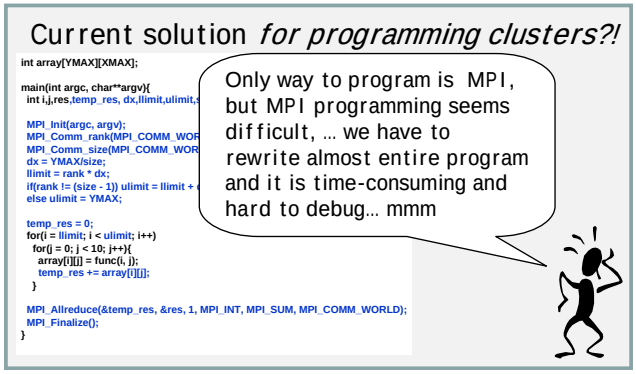
University of Tsukuba

What's XcalableMP?

Although MPI is a de-facto standard for parallel programming on distributed memory systems, writing MPI programs is often time-consuming and complicated process. XcalableMP is a directive-based language extension which allows users to develop parallel programs for distributed memory systems easily and tune the performance by having minimal and simple notations. So far, there have been a number of parallel programming languages for distributed memory architectures, including High Performance Fortran (HPF). However, these programming models have not been commonly used any more. On the other hand, OpenMP is widely used on shared memory architectures including SMP-configured PC clusters or multi-core CPUs. The most important feature of OpenMP from the perspective of programmability is to enable parallelization with simple directives that helps users extend their codes relatively easily from sequential ones. The target platform of OpenMP is however limited to shared memory architectures.

XcalableMP Application Program Interface (XcalableMP API) is a collection of compiler directives, runtime library routines that can be used to specify distributed-memory parallel programming in C and Fortran program. This specification provides a model of parallel programming for distributed memory multiprocessor systems, and the directives extend the C and Fortran base languages as to describe distributed memory parallel program, as in OpenMP.

XcalableMP introduces simple, but effective features to describe typical scientific applications using a concept similar to OpenMP. This paradigm features functions for array distribution and work mapping for loop on parallel processes, which are normally executed as MPI processes. These features can be coded using directives similar to OpenMP.



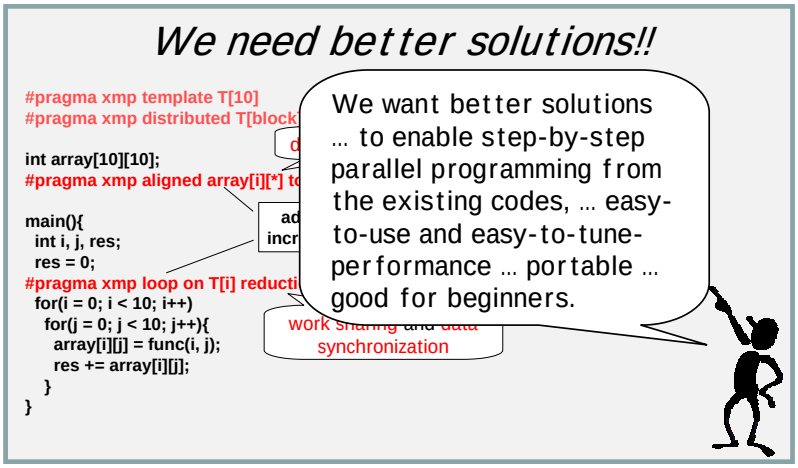
The specification has been being designed by XcalableMP Specification Working Group which consists of members from academia and research labs to industries in Japan.

Features of XcalableMP are summarized as follows:

- XcalableMP supports typical parallelization based on the data parallel paradigm and work mapping under "global view programming model", and enables parallelizing the original sequential code using minimal modification with simple directives, like OpenMP. Many ideas on "global-view" programming are inherited from HPF (High Performance Fortran).
- The important design principle of XcalableMP is "performance-awareness". All actions of communication and synchronization are taken by directives, different from automatic parallelizing compilers. The user should be aware of what happens by XcalableMP directives in the execution model on the distributed memory architecture.
- XcalableMP also includes CAF-like PGAS (Partitioned Global Address Space) feature as "local view" programming.
- Extention of existing base languages with directives is useful for rewriting cost and education cost. XcalableMP APIs are defined on C and Fortran 95 as a base language.
- For flexibility and extensibility, the execution model allows to combine with explicit MPI coding for more complicated and tuned parallel codes and libraries. For multi-core and SMP clusters, OpenMP directives can be combined into XcalableMP for thread programming inside each node as a hybrid programming model.(Under discussion)

XcalableMP

www.xcalablemp.org



Programming model of XcalableMP

The parallel execution model of XcalableMP is a Single Program Multiple Data (SPMD) model. As in MPI, all parallel processes in nodes start their execution from the same main function. When the thread encounters XcalableMP directives, the synchronization and communication occurs between nodes. That is, no synchronization and communications happen without directives. In this case, the program does duplicated execution of the same program on local memory in each node. As default, data declared in the program is allocated in each node, and is referenced locally by threads executed in the node.

The global-view programming model is useful when, starting from sequential version of the program; the programmer parallelizes it in data-parallel model by adding directives incrementally with minimum modifications. As these directives can be ignored as a comment by the compilers of base languages (C and Fortran), an XcalableMP program derived from a sequential program can preserve the integrity of original program when it is run sequentially.

The global-view programming model shares major concepts with HPF. ***Nodes directive*** declares a node array to express a set of nodes. The programmer describes the data distribution of data shared among the nodes by ***data distribution directives***. To specify the data distribution, the ***template*** is used as a dummy array distributed on nodes. A distributed array is declared by aligning the array to the template by ***align*** directive.

The work mapping in loop iteration is described by the ***loop directive*** as in OpenMP. ***Loop construct*** maps iterations to the node where referenced data is located. Template is used to specify the mapping of iteration. By using the same template used for the data distribution, iterations are assigned to the node of the data. It should be noted that in XcalableMP the programmer must control all data reference required computations done locally by any appropriate directives.

```
#pragma xmp nodes p(*)
#pragma xmp template t(0:N-1)
#pragma xmp distributed t(block) onto p
...
double w[N], p[N], q[N], r[N];
#pragma xmp align [i] with t(i):: p, q, r, w
#pragma xmp shadow [*] :: p
...
/* code fragment from conj_grad in NPB CG */
sum = 0.0;
#pragma xmp loop on t(j) reduction(+:sum)
    for (j = 1; j <= lastcol-firstcol+1; j++) {
        sum = sum + r[j]*r[j];
    }
    rho = sum;
    for (cgit = 1; cgit <= cgitmax; cgit++) {
#pragma xmp reflect p
#pragma xmp loop on t(j)
        for (j = 1; j <= lastrow-firstrow+1; j++) {
            sum = 0.0;
            for (k = rowstr[j]; k <= rowstr[j+1]-1; k++) {
                sum = sum + a[k]*p[colidx[k]];
            }
            w[j] = sum;
        }
#pragma xmp loop on t(j)
        for (j = 1; j <= lastcol-firstcol+1; j++) {
            q[j] = w[j];
        }
    …
  }
```

A fragment of code taken from NBP CG (one-dimensional parallelization) is shown in the box. ***Shadow directives*** are used to declare the shadow region of each array, which can be synchronized by ***reflect directive***..

In order to describe communication, the ***gmove construct*** is a powerful operation in global-view programming in XcalableMP: It copies data of a distributed array in global-view. This directive is followed by the assignment statement of scalar value and array sections.

In addition to the "global view" programming described above, XcalableMP also includes CAF-like PGAS (Partitioned Global Address Space) feature as “local view” programming. XcalableMP adopts coarray notations as an extension of languages for local view programming. In case of Fortran as the base language, most coarray notations are compatible to that of Coarray Fortran(CAF) expect that the task constructs are used for task parallelism. In order to use coarray notations in C, we propose some language extension of the language. A coarray is declared by the coarray directive in C.

XcalableMP Specification Working Group

Objectives and Mission

- Making a draft on XcalableMP parallel language for “standard” parallel programming.
- To propose the draft to “world-wide” community.

Members

Academia: M. Sato, T. Boku (compiler and system, U. Tsukuba), K. Nakajima (app. and programming, U. Tokyo), T. Nanri (system, Kyushu U.), Y. Okabe, M. Yasugi (HPF and compiler, Kyoto U.)

Research Lab.: M. Yokokawa (RIKEN), H. Sakagami (app. and HPF, NIFS), Y. Matsuo (app., JAXA), H. Uehara (app., JAMSTEC/ES)

Industries: H. Iwashita and K. Hotta (HPF and XPFortran, Fujitsu), H. Murai and S. Sakon (HPF, NEC), T. Anzaki and K. Negishi (Hitachi)

References

[1] H. Murai, T. Araki, Y. Hayashi, K. Suehiro and Y. Seo: Implementation and Evaluation of HPF/SX V2, Concurrency and Computation - Practice & Experience, Vol.14, No. 8-9, Wiley (2002), 603-629.

[2] J. Lee, M. Sato and T. Boku, "OpenMPD: A Directive-Based Data Parallel Language Extensions for Distributed Memory Systems", First International Workshop on Parallel Programming Models and Systems Software for High-End Computing (P2S2), 2008

This research is carried out as a part of “Seamless and Highly-productive Parallel Programming Environment for High-performance computing” project funded by Ministry of Education, Culture, Sports, Science and Technology, JAPAN.