XcalableMP

Directive-Based Language eXtension for Scalable and Performance-Aware Parallel Programming



T2K Open Supercomputer Alliance University of Tsukuba University of Tokyo Kyoto University

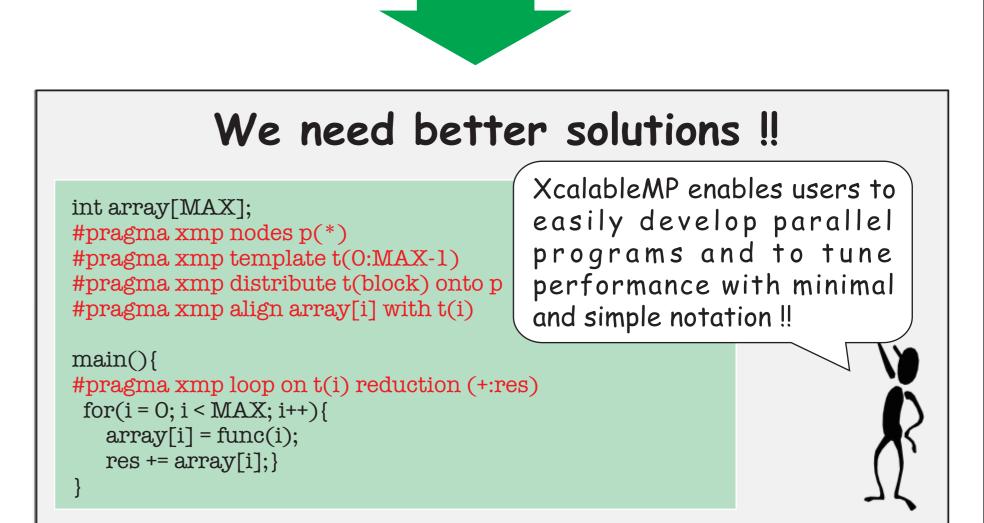
Overview

MPI is widely used as a parallel programming model. However, the programming cost of MPI is high.

XcalableMP is a directive-based language extension which allows users to easily develop parallel programs for distributed memory systems and to tune the performance by having minimal and simple notations.

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

Current Solution for parallel programming int array[MAX]; Only way to program is MPI, but MPI progrraming seems main(int argc, char * *argv){ MPI_Init(&argc, &argv); difficult,... we have to rerwite MPI_Comm_rank(MPI_COMM_WORLD, almost entire program and it MPI_Comm_size(MPI_COMM_WORLD, & dx = MAX/size;is time-consuming and hard to llimit = rank * dx; debug... mmm if(rank != (size -1)) ulimit = llimit + dx; else ulimit = MAX; $temp_res = 0;$ for(i=llimit; i < ulimit; i++){</pre> array[i] = func(i); temp_res += array[i];} MPI_Allreduce(&temp_res, &res, 1, MPI_INT, MPI_SUM, ...); MPI_Finalize();



Language Features

- XcalableMP supports typical parallelization based on the data parallel paradigm and work mapping under "global-view" programming model.
- The important design principle of XcalableMP is "performance-aware-ness". All actions of communication and synchronization are taken by directives, different from automatic parallelizing compilers.
- XcalableMP also includes CAF-like PGAS (Partitioned Global Address Space) feature as "local-view" programming.
- XcalableMP APIs are defined on C and Fortran 95 as a base language.

You can download XcalableMP from http://www.xcalablemp.org

Code Example (HPCC STREAM)

```
double a[SIZE], b[SIZE], c[SIZE];
#pragma xmp nodes p(*)
#pragma xmp template t(O:SIZE-1)
#pragma xmp distribute t(block) onto p
#pragma xmp align [i] with t(i) :: a,b,c
...
#pragma xmp loop on t(i)
for(i = 0; i < SIZE; i++) {
    a[i] = b[i] + scalar*c[i];
}</pre>
```