Center for Computational Sciences, University of Tsukuba
www.ccs.tsukuba.ac.jp

XcalableMP : Directive-Based Language eXtention for Scalable and Performance-Aware Parallel Pragramming

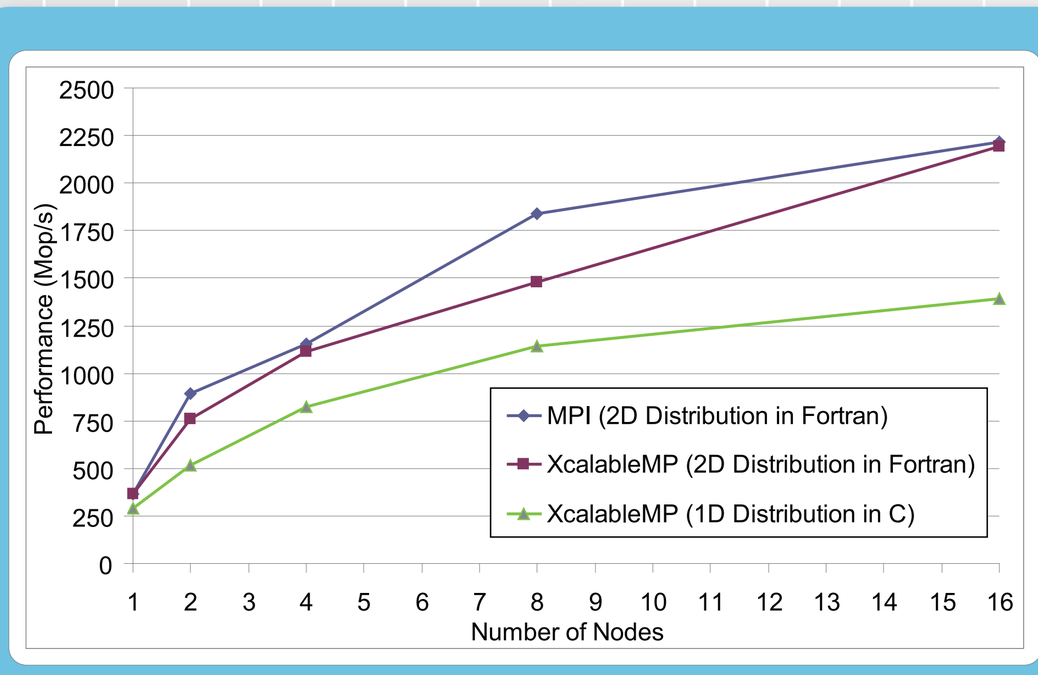
Although MPI is a de facto standard for parallel programming on distributed memory systems, writing MPI programs is often a time-consuming and complicated process. XcalableMP is a directive-based language extension which allows users to develop parallel programs for distributed memory systems easily and 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.

- 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 Peformance 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.
- XcalableMP APIs are defined on C and Fortran 95 as a base language.

Code Examples (NPB-CG)

```
!$XMP nodes proc(*)
!$XMP template (NA) :: t
!$XMP distribute (block) onto proc :: t
!$XMP align (j) with t(j) :: p, w
!$XMP shadow (*) :: p
. . .
!$XMP reflect p
!$XMP loop on t(j)
do j=1,lastrow-firstrow+1
   sum = 0.d0
   do k=rowstr(j),rowstr(j+1)-1
      sum = sum + a(k)*p(colidx(k))
   enddo
w(j) = sum
enddo
```

XMP



XcalableMP enables NPB-CG to be parallelized with 2D block distribution (equivalent to MPI version), and this result shows it achieves better performance than 1D block distribution.

Performance of XcalableMP Codes

```
do j=1,lastrow-firstrow+1
  sum = 0.d0
  do k = rowstr(j),rowstr(j+1)-1
    sum = sum + a(k)*p(colidx(k))
  enddo
  w(j) = sum
enddo
do i = l2npcols, 1, -1
  call mpi_irecv( q(reduce_recv_starts(i)), ... )
  call mpi_send(  w(reduce_send_starts(i)), ... )
  call mpi_wait( request, status, ierr )
  do j = send_start,send_start + reduce_recv_lengths(i) - 1
    w(j) = w(j) + q(j)
  enddo
enddo
if( l2npcols .ne. 0 )then
  call mpi_irecv( q, ... )
  call mpi_send(  w(send_start), ... )
  call mpi_wait( request, status, ierr )
else
  do j = 1,exch_recv_length
    q(j) = w(j)
  enddo
endif
```

MPI

```
!HPF$ independent
  do j=1,lastrow-firstrow+1
!HPF$ on home(w(j)), local(a,colidx) begin
  sum = 0.d0
  do k=rowstr(j),rowstr(j+1)-1
    sum = sum + a(k)*p(colidx(k))
  enddo
  w(j) = sum
!HPF$ end on
  enddo
  do j=1,lastcol-firstcol+1
    q(j) = w(j)
  enddo
```

HPF

```
! data exchange with CAF
! q(:) = w(:)
! synchronize data
call sync_notify(proc(i)+1)
call sync_wait(proc(i)+1)
! data exchange using co-array
q(n1:n2) = w(m1:m1+n2-n1)[proc(i)]
! synchronize communication
call sync_notify(proc(i)+1)
call sync_wait(proc(i)+1)
```

CAF

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.

www.xcalablemp.org