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

XcalableMP : directive-based language eXtention for Scalable and performance-tunable Parallel Pragramming

Although MPI is the 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 tune the performance by having minimal and simple notations.

- Extend existing base languages with directives for rewriting cost and education cost.
- XcalableMP supports typical parallelization based on the data parallel paradigm and work sharing under “global view”, and enables parallelizing the original sequential code using minimal modification with simple directives, like OpenMP.
- XcalableMP also includes CAF-like PGAS (Partitioned Global Address Space) feature as “local view” programming.
- Explicit communication and synchronization. All actions are taken by directives for being “easy-to-understand” in performance tuning.
- 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.

Code Examples (NPB-CG)

XMP

```
!$XMP distribute (block) :: p, w
!$XMP shadow (*) :: p
!$XMP reflect p
!$XMP loop affinity(w)
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
```

MPI

```
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
```

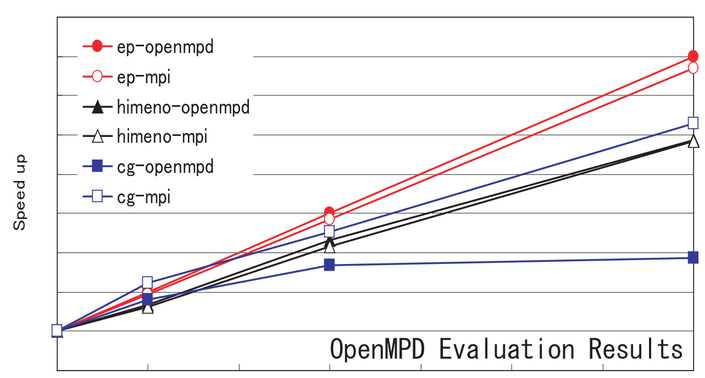
HPF

```
!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
```

CAF

```
! 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)
```

XcalableMP is being designed based on the experiences of HPF, Fujitsu XPF(VPP Fortran) and OpenMPD. The performance of XcalableMP code will be estimated by the almost equivalent code in OpenMPD



- execution environment

Item	Name
CPU	Intel Xeon 2.4Ghz
Memory	1GB
Network	1000BASE-T
MPI	LAM 7.0.6
OS	Linux 2.6.9

Jinpil Lee, Mitsuhisa Sato and Taisuke 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

XcalableMP is now under design. If you have any comments and requests, please contact to Prof. Mitsuhisa Sato (msato@cs.tsukuba.ac.jp). Your comments and contributions will be appreciated.

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