XcalableMP

$\langle$*ex-scalable-em-p*$\rangle$

Language Specification

Version 2.0 DRAFT

XcalableMP Specification Working Group

November, 2016

Contents

List of Figures

List of Tables

Acknowledgment

The specification of XcalableMP is designed by the XcalableMP Specification Working Group, which consists of the following members from academia, research laboratories, and industries.

• Tatsuya Abe	RIKEN
• Tokuro Anzaki	Hitachi
• Taisuke Boku	University of Tsukuba
• Toshio Endo	TITECH
• Yoshinari Fukui	JAMSTEC
• Yasuharu Hayashi	NEC
• Atsushi Hori	RIKEN
• Kohichiro Hotta	Fujitsu
• Hidetoshi Iwashita	RIKEN
• Susumu Komae	AXE
• Atsushi Kubota	Hiroshima City University
• Jinpil Lee	University of Tsukuba
• Toshiyuki Maeda	RIKEN
• Motohiko Matsuda	RIKEN
• Yuichi Matsuo	JAXA
• Kazuo Minami	RIKEN
• Shoji Morita	AXE
• Hitoshi Murai	RIKEN
• Kengo Nakajima	University of Tokyo
• Takashi Nakamura	JAXA
• Tomotake Nakamura	RIKEN
• Mamoru Nakano	CRAY
• Masahiro Nakao	RIKEN
• Takeshi Nanri	Kyusyu University
• Kiyoshi Negishi	Hitachi
• Satoshi Ohshima	University of Tokyo
• Yasuo Okabe	Kyoto University
• Hitoshi Sakagami	NIFS
• Tomoko Sakari	Fujitsu
• Shoich Sakon	NEC
• Mitsuhisa Sato	University of Tsukuba
• Taizo Shimizu	PC Cluster Consortium
• Takenori Shimosaka	RIKEN
• Yoshihisa Shizawa	RIST

• Shozo Takeoka	AXE
• Hitoshi Uehara	JAMSTEC
• Eiji Yamanaka	Fujitsu
• Masahiro Yasugi	Kyushu Institute of Technology
• Mitsuo Yokokawa	Kobe University

This work was supported by "Seamless and Highly-productive Parallel Programming Environment for High-performance Computing" project funded by Ministry of Education, Culture, Sports, Science and Technology, Japan, and is supported by PC Cluster Consortium.

Chapter 1

Overview of the XcalableMP Model and Language

1.1 Hardware Model

The target of XcalableMP is distributed-memory multicomputers (Figure 1.1). Each computation node, which may contain several cores, has its own local memory (shared by the cores, if any), and is connected with each other via an interconnection network. Each node can access its local memory directly and remote memory, that is, the memory of another node indirectly (i.e. via communication). However, it is assumed that accessing remote memory is much slower than accessing local memory.

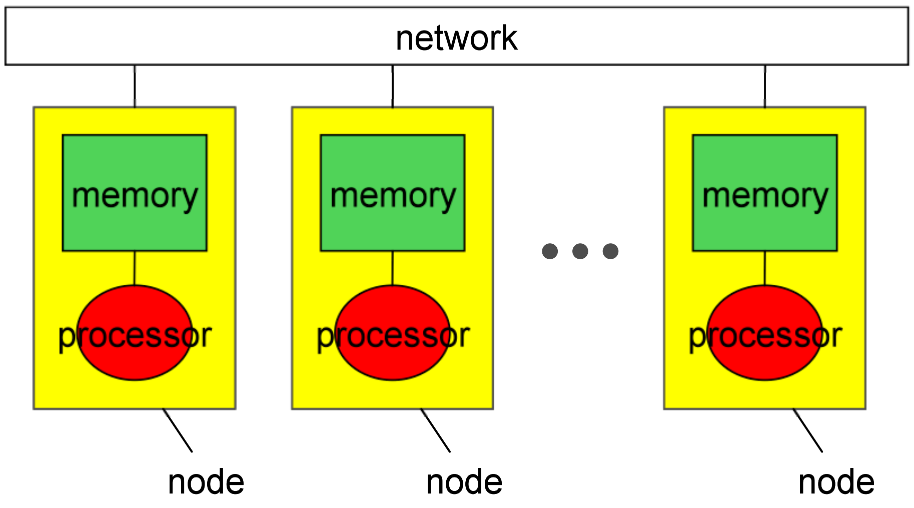

Figure 1.1: Hardware Model

1.2 Execution Model

The XcalableMP runtime system creates a team of threads on each node when starting the user program execution. This team of threads is composed by a single master thread and serveral additional worker threads.

An XcalableMP program execution is based on the Single Program Multiple Data (SPMD) model, where the master thread on each node starts execution from the same main routine and keeps executing the same code independently (i.e. asynchronously), as if an implicit tasklet region encloses the whole program, until it encounters an XcalableMP construct.

The set of nodes on each of which a thread executes a procedure, a statement, a loop, a block, etc. is referred to as its executing node set and determined by the innermost task, loop, or array directive surrounding it dynamically, or at runtime.

The current (executing) node set is the executing node set for the current context, which is managed by the XcalableMP runtime system on each node. The current node set at the beginning of the program execution, or the entire node set, is a node set that contains all the available nodes, which can be specified in an implementation-dependent way (e.g. through a command-line option).

When a thread encounters at runtime either a loop, array, or task construct, and is executed on a node in the node set specified by the on clause of the directive, it updates the current node set with the specified one and executes the body of the construct, after which it resumes the last executing node set and proceeds to execute the following statements.

Particularly when a thread encounters a loop or an array construct, it executes the loop body or the array assignment in parallel with threads on other nodes, so that each iteration of the loop or each element of the assignment is independently executed on the node where the specified data element resides.

When a thread encounters a synchronization or a communication directive, synchronization or communication might occur between it and threads on other nodes. That is, such global constructs should be performed collectively on the current nodes. Note that neither synchronizations nor communications occur without these constructs specified.

When a thread encounters a tasklet or a taskletloop construct, a new tasklet is generated on the node. Execution of explicitly generated tasklets is assigned to one of the threads on the node, subject to the thread ’ s availability to execute work. Thus, execution of the new tasklet could be immediate, or deferred until later according to the tasklet scheduling constraint and thread availability.

The tasklet scheduling constraint is as follows:

- A dependent tasklet shall not be scheduled until its task dependences are fulfilled.

1.3 Base Languages

The XcalableMP language specification is defined on Fortran, C, and C++ as base languages. More specifically, the base language of XcalableMP Fortran is Fortran 2008, that of XcalableMP C ISO is C99, and that of XcalableMP C++ is C++11.

1.4 Glossary

1.4.1 Node Terminology

node A logical entity managed by the XcalableMP runtime system, which has its own local memory and can communicate with each other, and on which one or more threads can execute.

1.4.2 Thread Terminology

thread An execution entity managed by the XcalableMP runtime system.

1.4.3 Tasklet Terminology

tasklet	A specific instance of executable code and its data environment, generated when a thread encounters a tasklet or taskletloop construct.

Chapter 2

Directives

2.1 Tasklet Constructs

2.1.1 `tasklet` Construct

Synopsis

Syntax

[F]   `!$xmp task on` {*nodes-ref* | *template-ref*}
      structured-block
      `!$xmp end task`

[C]   `#pragma xmp task on` {*nodes-ref* | *template-ref*}
      structured-block

Description

Restrictions

2.1.2 `taskletwait` Construct

Synopsis

Syntax

[F]   `!$xmp taskletwait`

[C]   `#pragma xmp taskletwait`

Description

2.1.3 `taskletbarrier` Construct

Synopsis

Syntax

[F]   `!$xmp taskletbarrier`

[C]   `#pragma xmp taskletbarrier`

Description

Index