

<http://omni-compiler.org/>

# Omni Compiler and XcodeML: An Infrastructure for Source-to- Source Transformation

MS03 Code Generation Techniques for HPC Earth Science Applications

**Mitsuhisa Sato**

**(RIKEN / Advanced Institute for Computational Science, Japan)**

**Co-Authors: Hitoshi Murai (AICS, RIKEN, Japan); Masahiro Nakao (AICS, RIKEN, Japan); Hidetoshi Iwashita (AICS, RIKEN, Japan); Jinpil Lee (AICS, RIKEN, Japan); Akihiro Tabuchi (University of Tsukuba, Japan)**

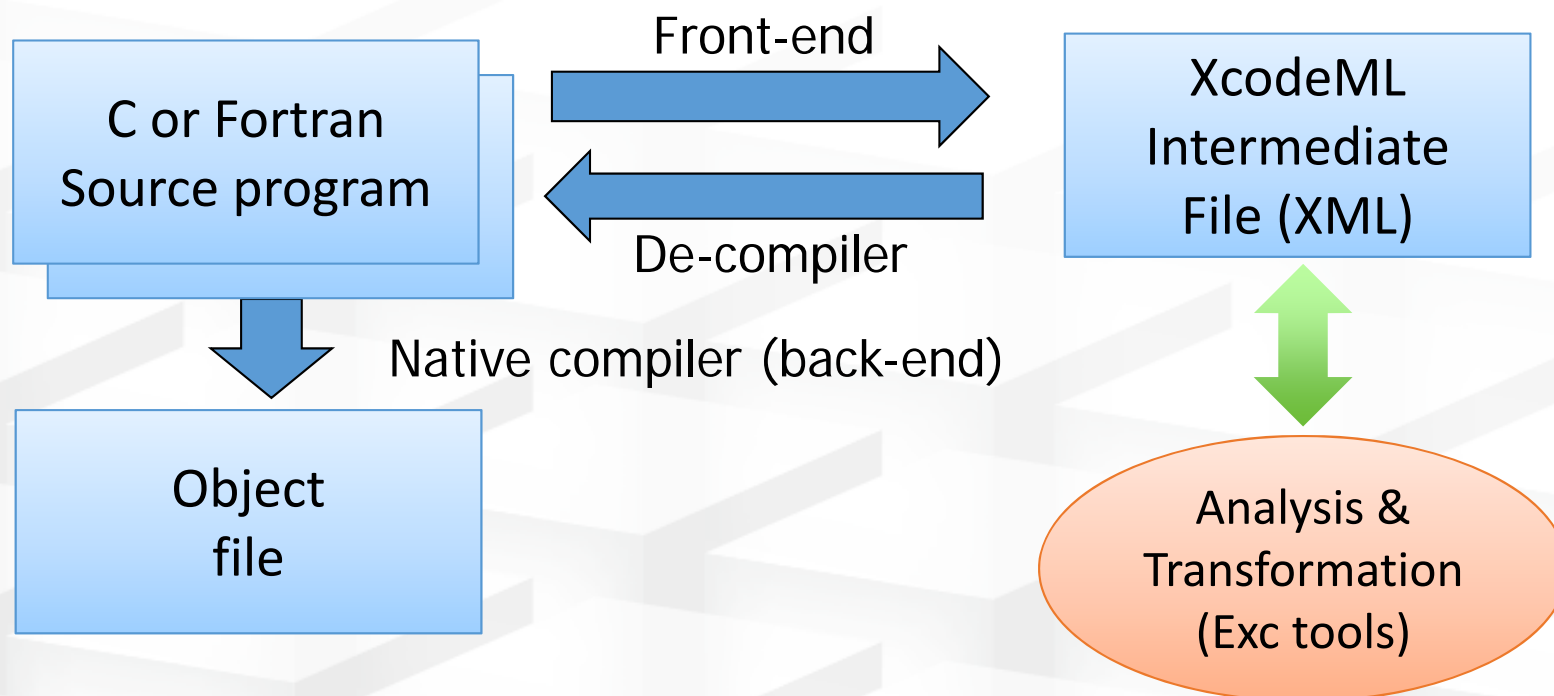
Wednesday, June 8, 2016

# What is “Omni Compiler”?

- **Omni compiler is a collection of programs and libraries that allow users to build code transformation compilers.**
- **Omni Compiler is to translate C and Fortran programs with directive-based extensions such as XcalableMP and OpenACC directives into codes compiled by a native back-end compiler linked with runtime libraries.**
- **Supported directives:**
  - OpenMP (C, F95)
  - OpenACC (C)
  - XcalableMP (C, F95)
  - XcalableACC (C)

# Overview of Omni Compiler

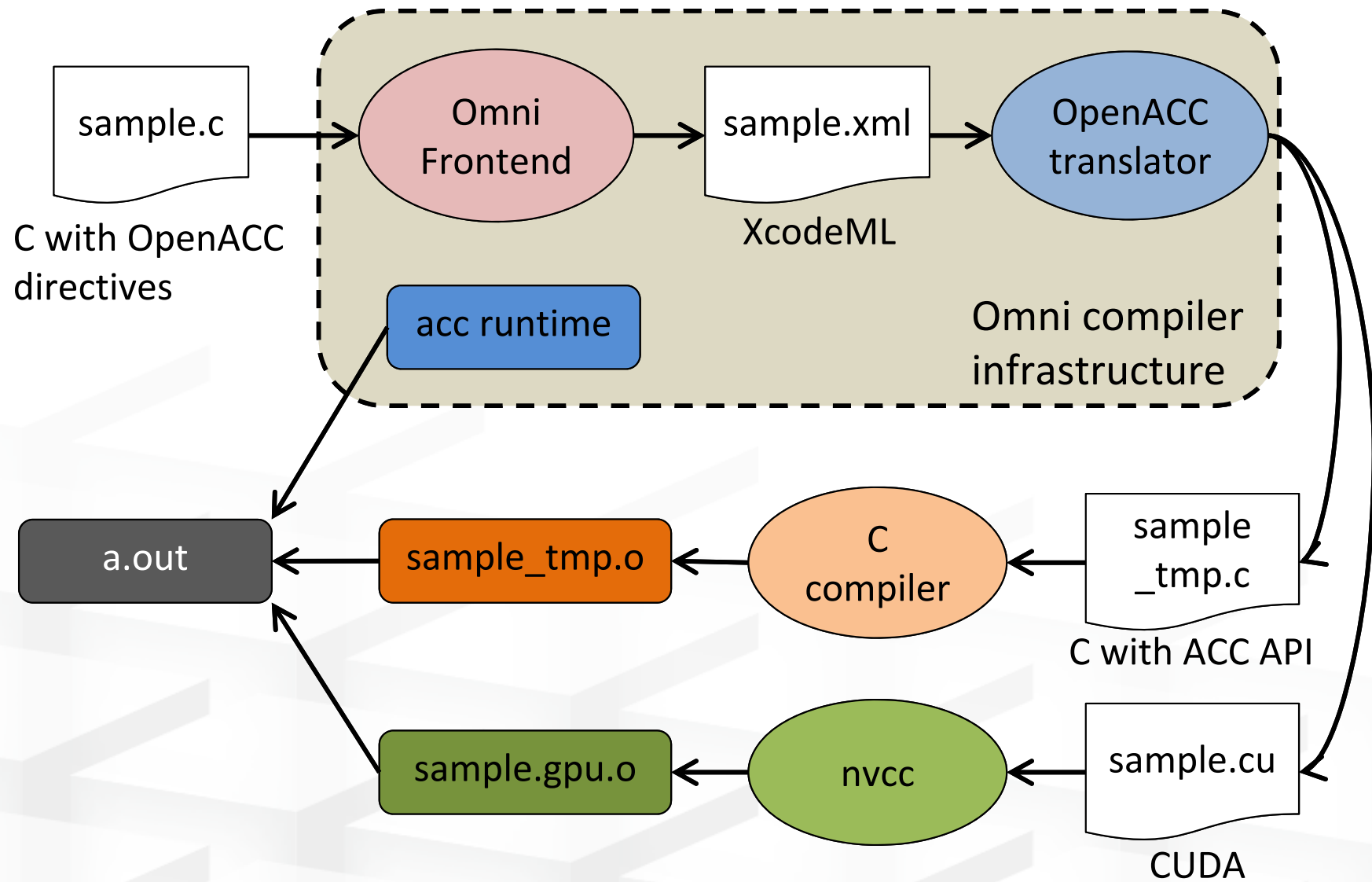
- The source programs in C and Fortran are translated into an XML intermediate form call XcodeML by the front-end programs, C-front and F-front.
  - Currently, only C and Fortran90 are supported.
- The XcodeML is an intermediate form designed to be translated back to the source program.
- Exc tools are a collection of Java classes to analyze and transform programs.



# Brief History of Omni Compiler Project

- **1999-2003: Omni OpenMP compiler project started at RWC (Real World Computing) project (1992-2002)**
  - OpenMP 1.0 (C and Fortran77 (f2c) ) supported
  - Cluster-enabled OpenMP (on top of DSM for Distributed memory)
  - Proprietary intermediate code (called Xcode) for C
- **(2003: supported by U. Tsukuba)**
- **2007: XcalableMP Specification Group launched**
- **2008-2011: e-science project**
  - C-front Revised (“gcc” based parser, gnu-extension supported)
  - F95 parser
  - XcodeML for C and F95
  - XcalableMP Spec v 1.0 released
  - Omni XcalableMP compiler for C released
- **2012 - : supported by RIKEN AICS, 2014- : adopted for post-K project**
  - Omni XcalableMP compiler for F95 released
  - Omni OpenACC (by U. Tsukuba)
  - 2016: Omni XcalableMP compiler v 1.0

# Example: Omni OpenACC Compiler



## ● XML for an intermediate code

- Human-readable format in XML
- It contains all information to translate the intermediate code back to the source program
- Support for C and Fortran 90 (currently, working also for C++)
- Syntax errors are supposed to be detected in front-end.
- Separate front-end and transformation phase, and back-end
  - Rich tools to handle XML can be used.

## ● Structures

- Type definitions
- Symbol tables for global declarations
- Program (function definitions) in ASTs

# Exc tool kits: A Java Class Library for Analysis and Transformation

- Currently, the tools consist of the following two Java package.
  - **Package exc.object:** this package defines the data structure. “Xobject”, which is an AST (Abstract Syntax Tree) for XcodeML intermediate form. It contains XcodeML input/output, classes and methods to build and access AST represented by “Xobjects”. It is useful to scan and analyze the program.
  - **Package exc.block:** this package defines data structure called “Block” to represents block structures of programs. It is useful to analyze and transform the programs.
- Classes to implement languages
  - Package exc.openmp: A translator for OpenMP compiler.
  - Package. exc.xcalablemp, exc.xmpf: A translator for XcalableMP compiler.

# Design patterns in exc tool kits (1)

- **XobjectIterator** is an iterator to traverse all Xobjects represented in Xobject class.
- **Subclass:**
  - bottomupXobjectIterator
  - topdownXobjectIterator

```
XobjectIterator i = new topdownXobjectIterator(x);  
for(i.init(); !i.end(); i.next(){  
    x = i.getXobject();  
    ...; // some operations ...  
}
```



## Design patterns in exc tool kits (2)

- **Class XobjectDef is a container of external definitions in Xobject File.**
- **Use visitor pattern to access XobjectDef in XobjectFile.**
  - iterateDef method of XobjectFile traverses XobjectDef.

```
class MyFuncVistor implements XobjectDefVisitor {  
    public void doDef(XobjectDef d){  
        ... do something on definition of d ...  
    }  
    ...  
}
```

```
/* in main routine */  
XobjectVisitor op = new MyFuncVisitor;  
f.iterateFuncDef(op);
```

# Simple Example using exc tools: print all variable reference

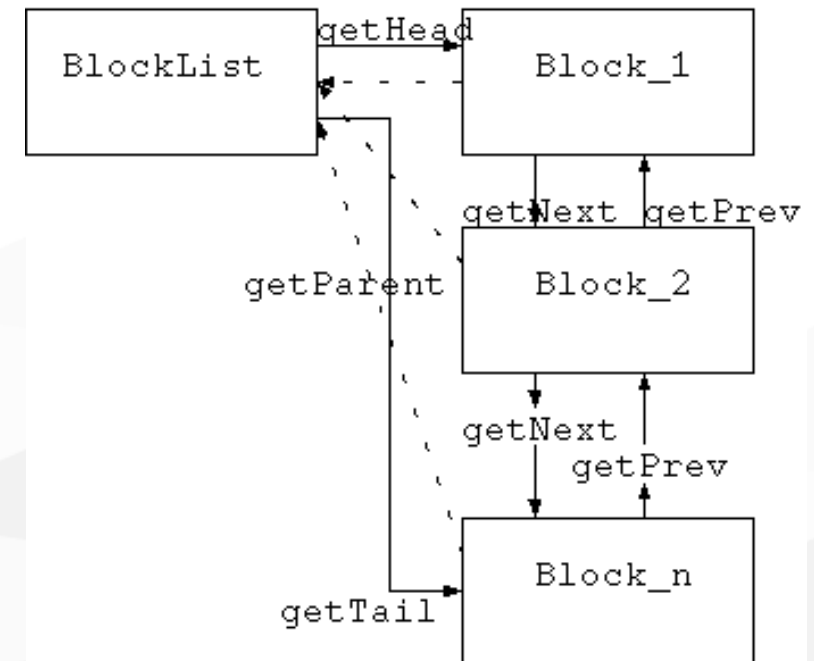
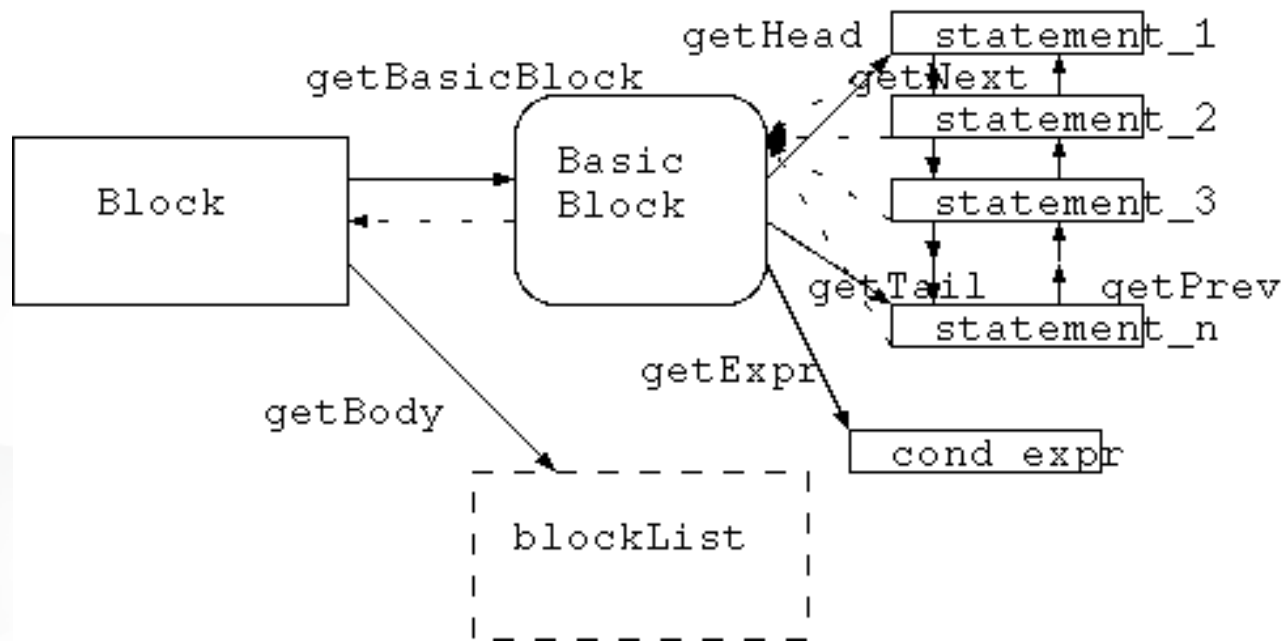
- Use XobjectIterator class to scan AST
- Use visitor pattern to access XobjectDef in XobjectFile.

```
class PrintVarRef implements XobjectDefVisitor {
    public void doDef(XobjectDef d){
        String fname = d.getName();
        XobjectIterator i = new topdownXobjectIterator(d.getFunBody());
        for(i.init(); !i.end(); i.next()){
            Xobject x = i.getXobject();
            if(x.isVariable() || x.isVarAddr())
                System.out.println("Variable '"+v.getName()+
                                   "' is referenced from Function '"+fname+"'");
        }
    }
}

public static void main(String args[]){
    XobjectFile f = new XobjectFile();
    f.Input("foo.x");
    XobjectVisitor op = new PrintVarName();
    f.iterateFuncDef(op);
}
```

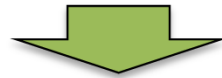
# Block and BlockList in exc.block

- **Data structure for code transformation**
  - Double-linked list to insert/delete.
  - Create new function definition



# Example 1: Translation of OpenACC parallel construct

```
#pragma acc parallel num_gangs(1) vector_length(128)
{
    /* codes in parallel region */
}
```



```
__global__ static void _ACC_GPU_FUNC_0_DEVICE( ... )
{
    /* codes in parallel region */
}
```

GPU kernel  
function

```
extern "C" void _ACC_GPU_FUNC_0( ... )
{
    dim3 _ACC_block(1, 1, 1), _ACC_thread(128, 1, 1);
    _ACC_GPU_FUNC_0_DEVICE<<<_ACC_block, _ACC_thread>>>( ..
. );
    _ACC_GPU_M_BARRIER_KERNEL();
}
```

kernel  
launch  
function

## Example 2: Translation of OpenACC loop construct

```
/* inner parallel region */  
#pragma acc loop vector  
for(i = 0; i < N; i++){  
    a[i]++;  
}
```

virtual index: `_ACC_idx`  
virtual index range : `_ACC_init, cond, step`

calculate the range of virtual index

```
/* inner gpu kernel code */  
int i, _ACC_idx;  
int _ACC_init, _ACC_cond, _ACC_step;  
_ACC_gpu_init_thread_x_iter(&_ACC_init, &_ACC_cond, &_ACC_step, 0, N, 1);  
for(_ACC_idx = _ACC_init; _ACC_idx < _ACC_cond; _ACC_idx += _ACC_step){  
    _ACC_gpu_calc_idx(_ACC_idx, &i, 0, N, 1);  
    a[i]++;  
}
```

virtual index

range variables

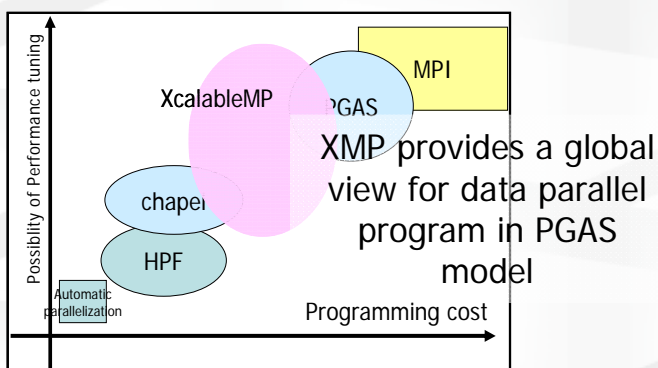
calculate 'i' from virtual index

loop body

# Projects using Omni Compiler/XcodeML

- **XcalableMP**
- **XcalableACC (XcalableMP + OpenACC)**
- **K-scope**

- What's XcalableMP (XMP for short)?
  - A PGAS programming model and language for distributed memory , proposed by **XMP Spec WG**
  - XMP Spec WG is a special interest group to design and draft the specification of XcalableMP language. It is now organized under **PC Cluster Consortium**, Japan. Mainly active in Japan, but open for everybody.
- Project status (as of Nov. 2014)
  - XMP Spec **Version 1.2** is available at XMP site. new features: mixed OpenMP and OpenACC , libraries for collective communications.
  - Reference implementation by U. Tsukuba and Riken AICS: **Version 0.93 (C and Fortran90)** is available for PC clusters, Cray XT and K computer. Source-to- Source compiler to code with the runtime on top of MPI and GasNet.
- **HPCC class 2 Winner 2013. 2014**



- Language Features
  - Directive-based language extensions for Fortran and C for PGAS model
  - Global view programming with global-view distributed data structures for data parallelism
    - SPMD execution model as MPI
    - pragmas for data distribution of global array.
    - Work mapping constructs to map works and iteration with affinity to data explicitly.
  - Rich communication and sync directives such as "gmove" and "shadow".
  - Many concepts are inherited from HPF
  - Co-array feature of CAF is adopted as a part of the language spec for local view programming (also defined in C).

## Code example

```
int array[YMAX][XMAX];
```

```
#pragma xmp nodes p(4)
#pragma xmp template t(YMAX)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i][*] to t(i)
```

data distribution

```
main(){
  int i, j, res;
  res = 0;
```

add to the serial code : incremental parallelization

```
#pragma xmp loop on t(i) reduction(+:res)
for(i = 0; i < 10; i++){
  for(j = 0; j < 10; j++){
    array[i][j] = func(i, j);
    res += array[i][j];
  }
}
```

work sharing and data synchronization

# XcalableACC(ACC) = XcalableMP+OpenACC

## ● Extension of XcalableMP for GPU

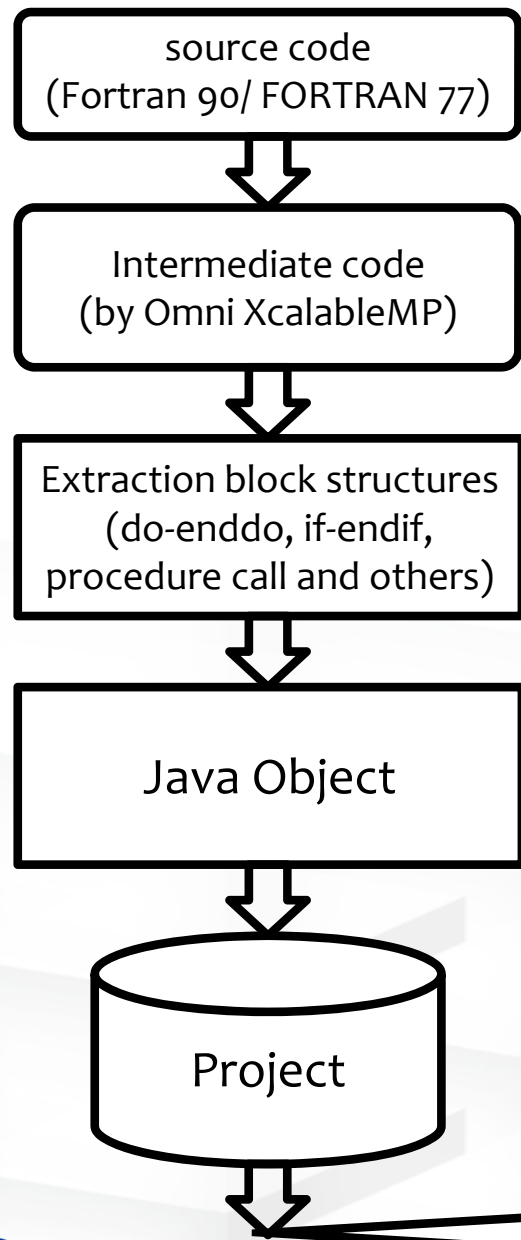
- A project of U. Tsukuba leaded by Prof. Taiuske Boku
- “vertical” integration of XcalableMP and OpenACC
  - Data distribution for both host and GPU by XcalableMP
  - Offloading computations in a set of nodes by OpenACC
- Proposed as unified parallel programming model for many-core architecture & accelerator
  - GPU, Intel Xeon Phi
  - OpenACC supports many architectures

Source Code Example: NPB CG

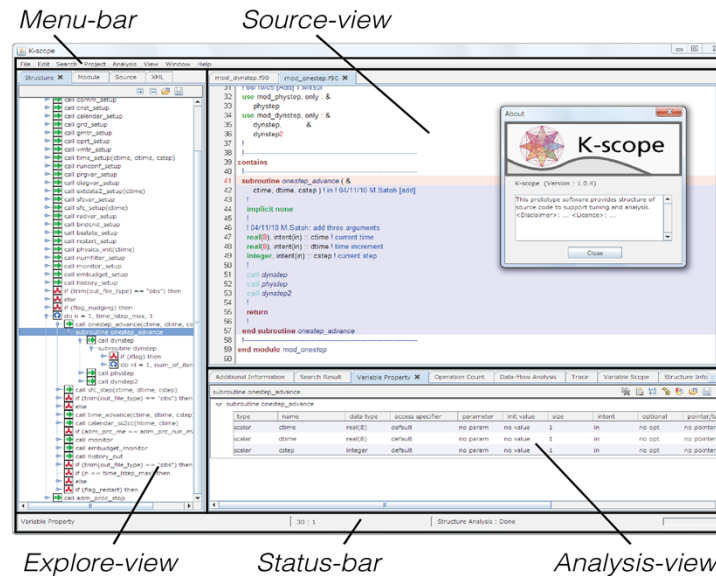
```
#pragma xmp nodes p(NUM_COLS, NUM_ROWS)
#pragma xmp template t(0:NA-1,0:NA-1)
#pragma xmp distribute t(block, block) onto p
#pragma xmp align w[i] with t(*,i)
#pragma xmp align q[i] with t(i,*)
double a[NZ];
int rowstr[NA+1], colidx[NZ];
...
#pragma acc data copy(p,q,r,w,rowstr[0:NA+1]¥
                        , a[0:NZ], colidx[0:NZ])
{
    ...
    #pragma xmp loop on t(*,j)
    #pragma acc parallel loop gang
    for(j=0; j < NA; j++){
        double sum = 0.0;
        #pragma acc loop vector reduction(+:sum)
        for (k = rowstr[j]; k < rowstr[j+1]; k++)
            sum = sum + a[k]*p[colidx[k]];
        w[j] = sum;
    }
    #pragma xmp reduction(+:w) on p(:,*) acc
    #pragma xmp gmove acc
    q[:] = w[:];
    ...
} //end acc data
```



# K-scope: static code analyzer

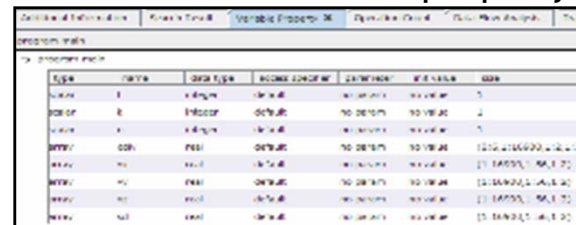


## Screenshot for interface



## Analysis-view

### List view for Variable property



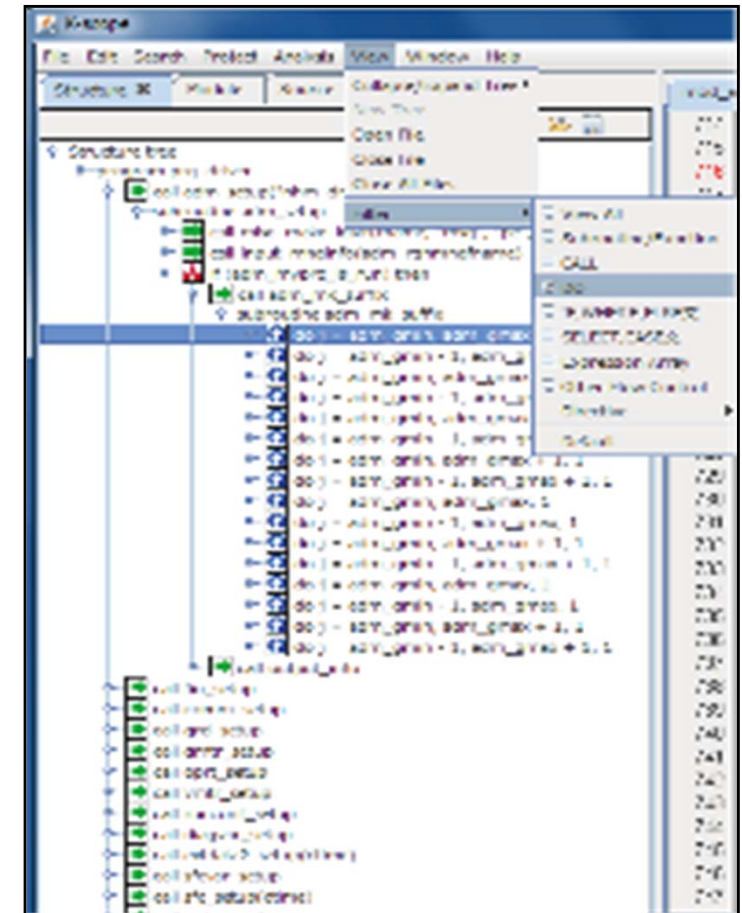
Variable	DATA TYPE	SCOPE	DEFINITION	INITIAL VALUE	ADDR
program main					
time	real	global	no default	no value	5
dim	integer	default	no default	no value	2
time	real	global	no default	no value	5
time	real	global	no default	no value	1210.1166000000000
time	real	global	no default	no value	1.1660000000000000
time	real	global	no default	no value	1.1660000000000000
time	real	global	no default	no value	1.1660000000000000
time	real	global	no default	no value	1.1660000000000000

### List view for Variable declaration/definition/reference



FOCUS

## Visualization for a program structure with form of a tree in the explore-view



# FLAGSHIP 2020 Project

## ● Missions

- Building the Japanese national flagship supercomputer, Post K, and
- Developing wide range of HPC applications, running on Post K, in order to solve social and science issues in our country.

## ● Planned Budget

- 110 Billion JPY (about 0.91 Billion USD at the rate 120 JPY/\$)
- including research, development (NRE) and acquisition/deploy, and application development

## ● Post K Computer: System and Software

- RIKEN AICS is in charge of development
- Fujitsu is selected as a vendor partner
- Started from 2014

**XcalableMP 2.0 is a language project in FS2020!!!**

CY	2014				2015				2016				2017				2018				2019				2020			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
	Basic Design				Design and Implementation								Manufacturing, Installation, and Tuning								Operation							

# Current Status and Plan

- **On going:**

- **Documentation !!!!!**

- Fortran 2003 support

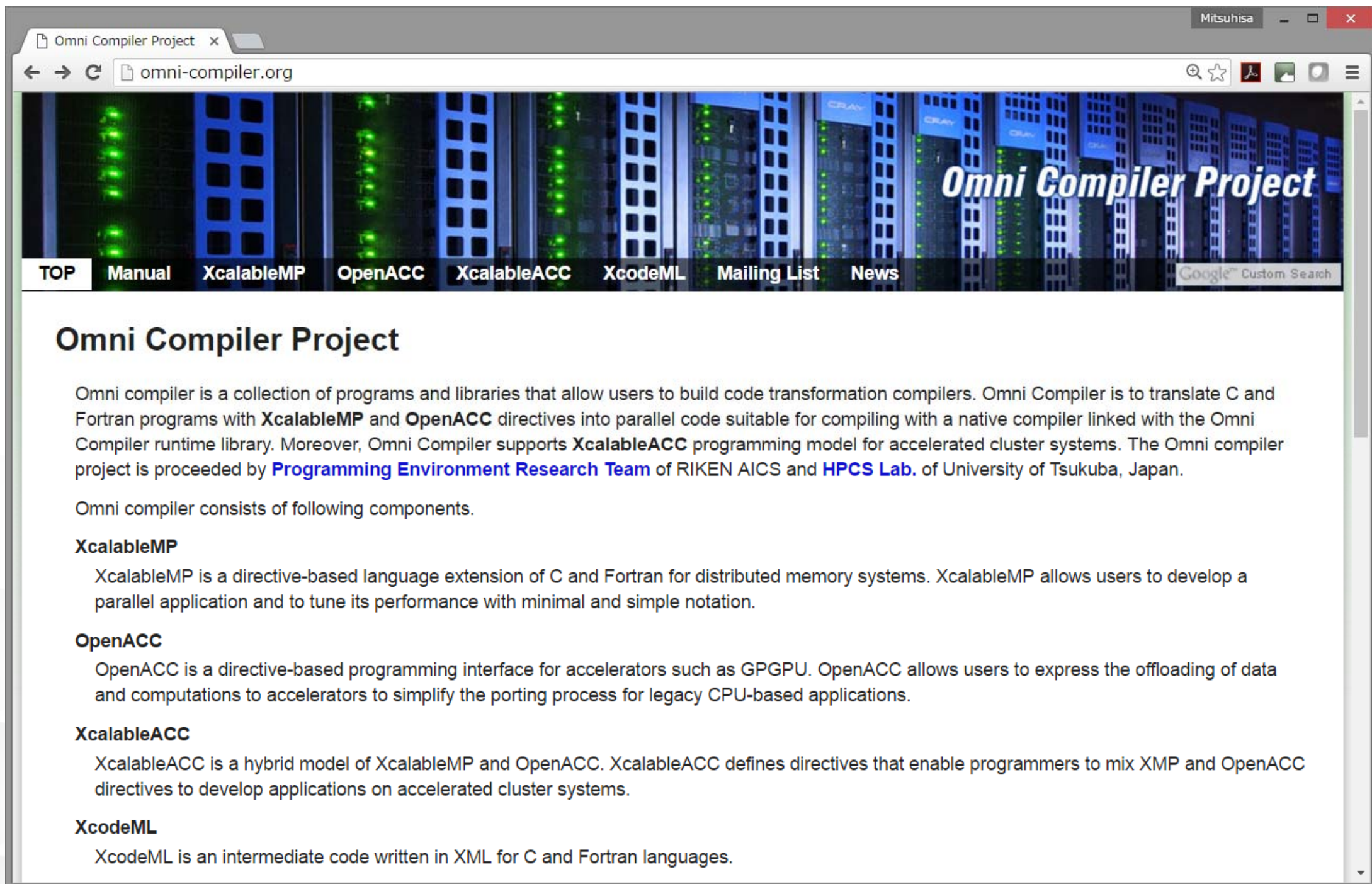
- We got support from the vender partner. They provide us test-suite of F2003.

- C++ support

- Using Clang/LLVM front-end to convert Clang AST to XcodeML

- **Plan**

- XMP 2.0 (tasklet construct for multithreaded execution in manycore)
  - XcalableMP for C++ (object-oriented features)
  - OpenACC (XcalableACC) for Fortran 2003

A screenshot of a web browser displaying the Omni Compiler Project website. The browser's address bar shows 'omni-compiler.org'. The website features a header with a background image of server racks and the text 'Omni Compiler Project'. Below the header is a navigation menu with links: TOP, Manual, XcalableMP, OpenACC, XcalableACC, XcodeML, Mailing List, and News. A Google Custom Search bar is also present. The main content area has a heading 'Omni Compiler Project' followed by a paragraph describing the project's purpose and its affiliation with RIKEN AICS and HPCS Lab. Below this, a section titled 'Omni compiler consists of following components.' lists four sub-projects: XcalableMP, OpenACC, XcalableACC, and XcodeML, each with a brief description of its functionality.

Omni Compiler Project

TOP Manual XcalableMP OpenACC XcalableACC XcodeML Mailing List News

## Omni Compiler Project

Omni compiler is a collection of programs and libraries that allow users to build code transformation compilers. Omni Compiler is to translate C and Fortran programs with **XcalableMP** and **OpenACC** directives into parallel code suitable for compiling with a native compiler linked with the Omni Compiler runtime library. Moreover, Omni Compiler supports **XcalableACC** programming model for accelerated cluster systems. The Omni compiler project is proceeded by [Programming Environment Research Team](#) of RIKEN AICS and [HPCS Lab.](#) of University of Tsukuba, Japan.

Omni compiler consists of following components.

### XcalableMP

XcalableMP is a directive-based language extension of C and Fortran for distributed memory systems. XcalableMP allows users to develop a parallel application and to tune its performance with minimal and simple notation.

### OpenACC

OpenACC is a directive-based programming interface for accelerators such as GPGPU. OpenACC allows users to express the offloading of data and computations to accelerators to simplify the porting process for legacy CPU-based applications.

### XcalableACC

XcalableACC is a hybrid model of XcalableMP and OpenACC. XcalableACC defines directives that enable programmers to mix XMP and OpenACC directives to develop applications on accelerated cluster systems.

### XcodeML

XcodeML is an intermediate code written in XML for C and Fortran languages.