234Compositor Library

User's Manual

February 2015

7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, 650-0047, JAPAN.
http://www.aics.riken.jp/

Contents

Chapter 1

About this document

This document is the user manual for 234Compositor library.

1.1 About 234Compositor Library

234Compositor is a parallel image composition library being developed to perform sort-last image composition on massively parallel environments. It was designed to be called by parallel rendering applications in order to merge the rendered images into a single final image. 234Compositor receives the image data and some composition parameters in order to perform the image composition. It is based on 2-3-4 Decomposition for converting arbitrary number of nodes into power-of-two number of nodes, and in this initial version, Binary-Swap image composition algorithm was used to perform the main image composition. The final image gathering process can be performed through traditional MPI_Gatherv as well as the more scalable MPI_Gather with the cost of additional processing and memory overhead. However, the MPI_Gather version is recommended to be used for parallel image composition when using more than thousands of composition nodes. It has been designed to run on multi-core distributed systems, and it requires MPI and OpenMP libraries for building the library.

1.2 Format

The following format represents a Shell command:

$ command (command parameters)

or

 # command (command parameters)

A command that starts with "$" is to be executed by a user. A command that starts with "#" is to be executed by the administrator (root user in most cases).

1.3 Supported Environments

234Compositor Library has been tested in following environments:

- Linux / OpenMPI Compiler
- K Computer / Fujitsu Cross Compiler

Chapter 2

Building 234Compositor Library

2.1 Package Structure

The 234Compositor library package is stored in a file with the following name format:

234Compositor-x.y.tar.gz (where *x.y* designates the version)

When the file is expanded, the following directories and files should appear:

234Compositor-x.y
- *src*
- *include*
- *lib*
- *doc*
- *sample*

src/	It will contain the source files.
include/	It will contain the header files including those to be included by the rendering applications. The contents of this directory will be installed in *$prefix/include* after the *make install* command.
lib/	The libraries will be created and installed at *$prefix/lib* after the *make install* command.
doc/	It will contain the documents regarding 234Compositor.
sample/	It will contain some example codes for testing the 234Compositor.

2.2 Building the 234Compositor Library Package

234Compositor library offers two ways for package building differing how the environment variables are set: automatic setting through the configure script (*./configure* command) or manual setting. The following guide describes how to build the package using the configure script, and then using manual environment configuration.

2.3 Building using the Configure Script

It uses the shell environment to build the library package. The syntax for setting the environmental variables may differ depending on the utilized shell. Therefore, replace the environmental variables settings according to the utilized computer environment. In the following example, a working directory is created where the files are extracted and decompressed. The library is the built and installed into this working directory.

1. Create a working directory (called "work" herein) for the 234Compositor library package.

    $ mkdir work

    $ cp [package path] work

2. Change to the working directory and unzip the package.

    $ cd work

    $ tar zxf 234Compositor-x.y.tar.gz

3. Change to the 234Compositor directory generated by this file extracting process.

    $ cd 234Compositor-x.y

4. Execute the configure script by specifying appropriate configuration parameters.

    $ ./configure [configuration parameters]

    The configure script can be configured according to the utilized environment by choosing appropriate parameters. For further details on the available configuration parameters, see the next subsection (Section 2.4). After the execution, a *Makefile* suitable for the utilized environment will be created in the working directory.

5. Execute a make command to build the 234Compositor library.

    $ make

    This will create the following file: **src/lib234comp.a**

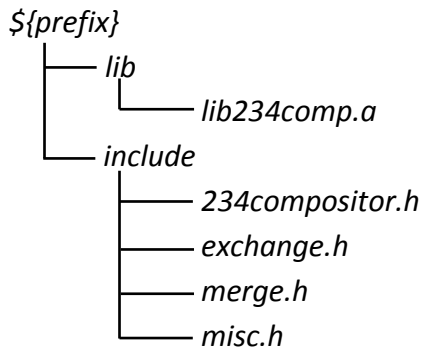
If you want to rebuild the 234Compositor library, delete the created files by the aforementioned *make* command by using the following command:

$ make clean

If you want to rerun the configure script or recreate the *Makefile*, then execute the following command in order to delete all information and restart from the configure script.

$ make distclean

6. Call the *make install* command in order to install the 234Compositor library and the header files for integration with the parallel rendering applications. These files will be installed at the directory specified with a *--prefix* option in the configure script:



If you have sufficient file permission to write under the installation directory, execute the following command:

$ make install

If administrator rights are required, the use the *sudo* command as follows:

$ sudo make install

If you are not allowed to use *sudo*, then log in as a *root* user and execute the *make install*.

$ su

(enter password)

 # make install

 # exit

To uninstall 234Compositor, use one of the three commands depending on your permission:

$ make uninstall

$ sudo make uninstall

 # make uninstall

2.4 Configure Script Options

Specify the install directory, compiler, MPI and OpenMP options as follows:

- **--prefix = INSTALL_DIR**
  Specify where to install the 234Compositor library and header files. For instance, if "*--prefix=/usr/local"* is specified, they will be installed in the following directories:
  Library: /usr/local/lib
  Header files: /usr/local/include

  If this option is not specified, "/usr/local/" will be used set as the installation directory.

- **--with-ompi = OPENMPI_DIR**
  Specify the path for the OpenMPI library being used for compilation.

- **--host = hostname**
  Specify the hostname in the case of cross compilation.

- **CC = CC_COMPILER**
  Specify the MPI enabled C compiler. 234Compositor has been tested with following compilers:
  - mpicc (GNU C compiler with OpenMPI library)
  - mpifccx (Fujitsu C cross-compiler with MPI library)

- **CFLAGS = C_COMPILER_OPTIONS**
  Specify the compiler options such as optimizations (-O3), the use of MPI_Gatherv (-D _GATHERV), and others.
  The recommended compiler option for GNU C compiler is:
  - **CFLAGS = -O3 –std=gnu99 –Wall –fopenmp [–D _GATHERV]**

  The recommended compiler option for K Computer and Fujitsu FX-10 is:
  - **CFLAGS = -Kfast,openmp –Xg –std=c99**

  For instance, a sample configure command when using GNU C and MPI_Gatherv might be:
  ./configure --prefix=/usr/local/ --with-ompi=/opt/openmpi CC=mpicc CFLAGS="-O3 -std=gnu99 -Wall -D _GATHERV -fopenmp"

2.5 Manually Building 234Compositor Library

This section and the following subsection will describe how to manually build 234Compositor library. If 234Compositor library is built manually, the "lib234comp.a" static library file will be generated. To utilize this library, the following files should be copied into a proper directory.

Library file:
lib/lib234comp.a
Include files:
include/234compositor.h
include/exchange.h
include/merge.h
include/misc.h

2.5.1 Editing the Makefile for Manual Configuration

For manual configuration, sample *Makefile* named *"Makefile_hand"* is available in the root directory.

- Makefile_hand
- Makefile_hand.K (for K computer)

Modify the *Makefile_hand* file according to the system and environment being used such as:

CC = (C compiler with MPI library)
CFLAGS = (Compiler options)
OMPFLAGS = (OpenMP library)

2.5.2 Make using Manually Configured Makefile

To build the library using manually configured *Makefile*, then use the following command:

$ make if Makefile_hand(.K)

This will generate the "lib234comp.a" library in "/lib" subdirectory.

Chapter 3

Using 234Compositor Library

The following sections describe the use of 234Compositor library API and some sample codes present in the *sample* folder.

3.1 234Compositor API

234Compositor library has been designed to be called by the parallel rendering application and it consists of three main functions:

- Init_234Composition (Initialize some variables and allocate memory)
- Do_234Composition (Execute the image composition)
- Destroy_234Composition (Destroy the variables and release the allocated memory)

3.1.1 Init_234Composition

Before calling the image composition command, there is a need to initialize some variables and allocate a buffer memory for temporary store of image data fragments. The parameters for calling the initialization function are:

Variable	Type
My Rank Number	unsigned int
Number of Nodes	unsigned int
Image Width	unsigned int
Image Height	unsigned int
Pixel ID	unsigned int

The pixel ID represents the pixel type of the image being composited, and although there are plenty of pixel types, in this initial version the 234Compositor has implemented the following types:

Pixel ID	Pixel Type
ID_RGBA32	32-bit RGBA (4 x 8-bit color and alpha components)
ID_RGBAZ64	64-bit RGBAZ (4 x 8-bit color and alpha components, and 32-bit depth component)
ID_RGBA128	128-bit RGBA (4 x 32-bit color and alpha components)
ID_RGBAZ160	160-bit RGBAZ (5 x 32-bit color, alpha and depth components)

3.1.2 Do_234Composition

Although the existence of different kinds of composition methods, in this initial version only the image composition based on back-to-front alpha blending has been implemented. For this purpose, it is required that the images to be composited are ordered correctly before calling the composition function. The assumption is that the images are sequentially ordered based in the MPI rank order, that is, as the MPI rank increases the images from the closest moves to the farthest from the viewpoint (Rank 0 over Rank 1 over … over Rank n).

The required parameters for calling the image composition are:

Variable	Type
My Rank Number	unsigned int
Number of Nodes	unsigned int
Image Width	unsigned int
Image Height	unsigned int
Pixel ID	unsigned int
Merge ID	unsigned int
Image Data	unsigned int* or float*
MPI Communicator	MPI_COMM

The merge ID represents the pixel merging method, and in this version only the following method has been implemented:

Merge ID	Pixel Merging Method
ALPHA_BtoF	Alpha Blending (Back-to-Front order)

In order to use the 234Compositor, the parallel rendering application should include the *"234compositor.h"* header file. In this file is also specified the pixel type and pixel merging methods available for using by the rendering application. Although some other pixel types and merging methods have already been defined in this header file, they are not functional and were just defined following the developmental roadmap.

#include "234compositor.h"

Once the initialization is called, and until the image size (width * height in Mega Pixels) remains within the following pre-defined threshold, there is no need to call the initialization again and successive image composition call can be executed:

- 4, 8, 16, 32, 64 and 128 Mega Pixels

3.1.3 Destroy_234Composition

At the end of image composition call or before calling the initializing function, it is recommended to release the variables and memory regions used by the 234Compositior Library. For this purpose, the *Destroy* command with following parameter should be called.

Variable	Type
Pixel ID	unsigned int

3.2 Sample Codes

3.2.1 No Graphics Library Requirements

Some sample codes has been placed in *"sample"* folder in order to check the functionality of 234Compositor library. In the root directory of "sample" folder, there are two sample codes where each rendering node generates partial Mandelbrot images to be composited by the composition nodes.

- **test_234byte_mandel.c** (Render and composite "unsigned int *" image data)
- **test_234float_mandel.c** (Render and composite "float *" image data)

The "compile.sh" file has an example for command line compilation and linking to produce the executables: **test_234byte_mandel** and **test_234float_mandel**.

The following command can be used to execute the test code, and appropriate modification should be applied depending on the working environment.

mpirun –np (number of nodes) test_234byte_mandel [OPTION]

where the OPTION is to write (1) or not (0) the input images (rendered images) as files.
The output is a raw image file named **output_XxY.rgba32** (X and Y represents the image width and height respectively). This raw image can be viewed by a sample program *"show_image"* placed in *"GLUT"* folder, and requires GLU and GLUT graphics library to be compiled.

In the case of **test_234float_mandel**, the output image will be named as **output_XxY.rgba128**.

3.2.2 With Graphics Library Requirements

If the running environment possesses *"GLU"* and *"GLUT"* graphics library, the test codes available in the *"GLUT"* library can be used. They are offline image compositor which loads pre-generated raw images stored as files and composite and outputs it as another image file. The test codes are as follows:

- **test_234byte.c** (Load and composite "unsigned int *" image data)
- **test_234float.c** (Load and composite "float *" image data)

The "compile.sh" file has an example for command line compilation and linking to produce the executables: **test_234byte** and **test_234float**.

It loads the image files presented at *"input_images"* folder where the images to be read are **XXXX.rgba32** or **XXXX.rgba128** where XXXX represents the sequential number to be read by the corresponding MPI rank node (0000 for the MPI Rank 0, 0001 for the MPI Rank 1, and so on).

The following two codes can be used for generating and checking the aforementioned images, and requires GLU and GLUT libraries (-lGLU –lglut) for linking:

- **generate_image.c** (Generate raw image files)
- **show_image.c** (GLUT-based raw image viewer)

The parameters for generating raw images (**generate_image**) are:

- Image Width
- Image Height
- Pixel ID
- Number of Images to be Generated
- Optional parameters
    - Alpha Value (from 0 to 255): Default is 127 (50%)
    - Generate in Big-Endian mode (Yes[1] or No[0]) : Default is "No"

There is a need to press "w" (WRITE) in order to start the image file generation, after calling the generate image with appropriate parameters.

The parameters for verifying the generated raw images (**show_image**) are:

- Image Width
- Image Height
- Optional parameters
    - Big-Endian image (Yes[1] or No[0]) : Default is "No"