NEXTGIS
OPENSOURCE GEOSPATIAL SOLUTIONS

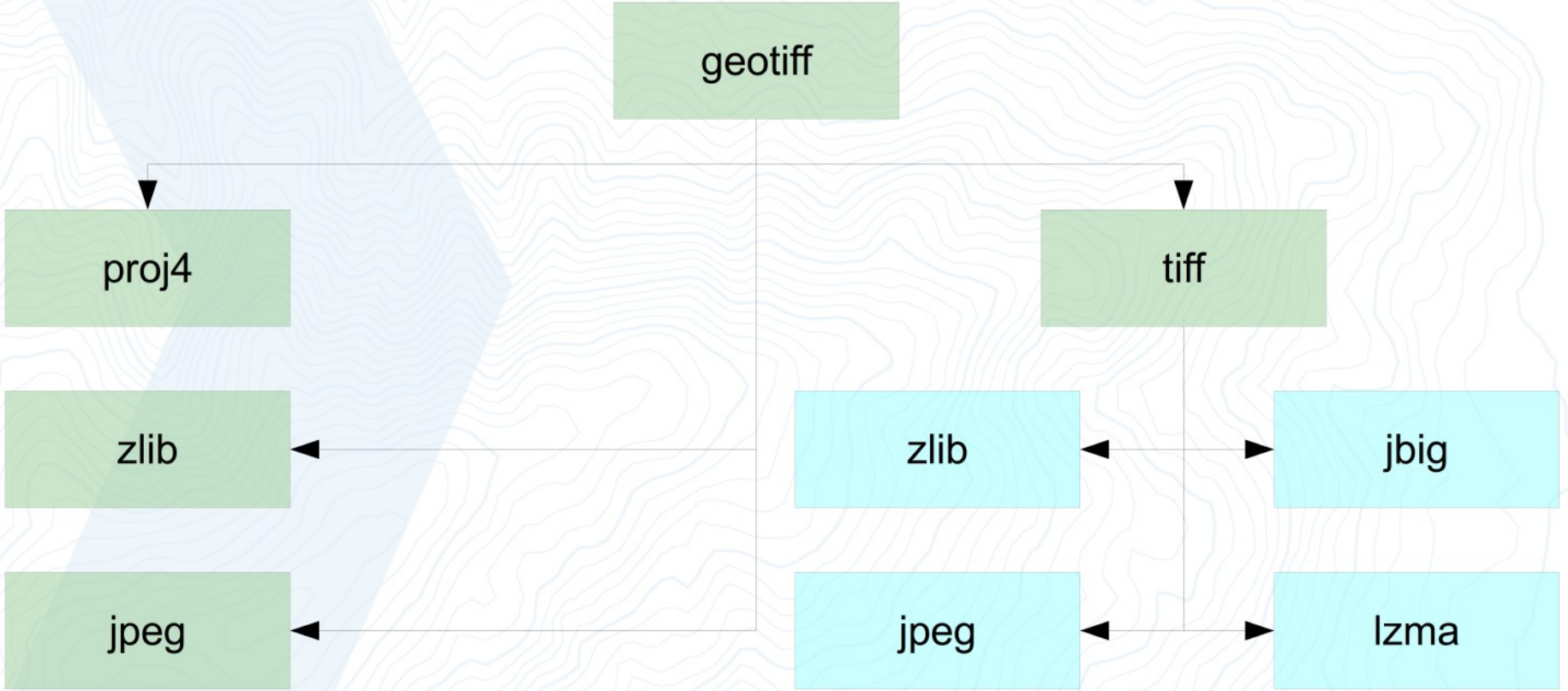
Borsch: modern build system for C/C++ GIS projects

Dmitry Baryshnikov

info@nextgis.com www.nextgis.com @GisBishop

Problem

Problem



Ideal build system

1. Crossplatform
2. Simple library inclusion
3. Truck library updates
4. Automatic dependency resolution
5. Build options and input parameters transferring from project to dependent libraries
6. Simple library preparation to use in build system

Build systems

- **make/autoconf**
- **CMake**
- Scons
- GYP
- Gradle
- Premake etc.

NextGIS Borsch

URL: **https://github.com/nextgis-borsch**

- Approach is close to original CMake ExternalProject
- Dependencies copied to a subfolder there build occurs
- The build system use Github
- To activate build system you only need three files:

FindAnyProject.cmake

FindExtProject.cmake

ExternalProject.cmake

Include library via NextGIS Borsch

1. Set additional path to search modules

   set(CMAKE_MODULE_PATH ${CMAKE_SOURCE_DIR}/cmake ${CMAKE_MODULE_PATH})

2. Include main file

   include(FindAnyProject)

3. Specify necessary libraries

   find_anyproject(TIFF REQUIRED)

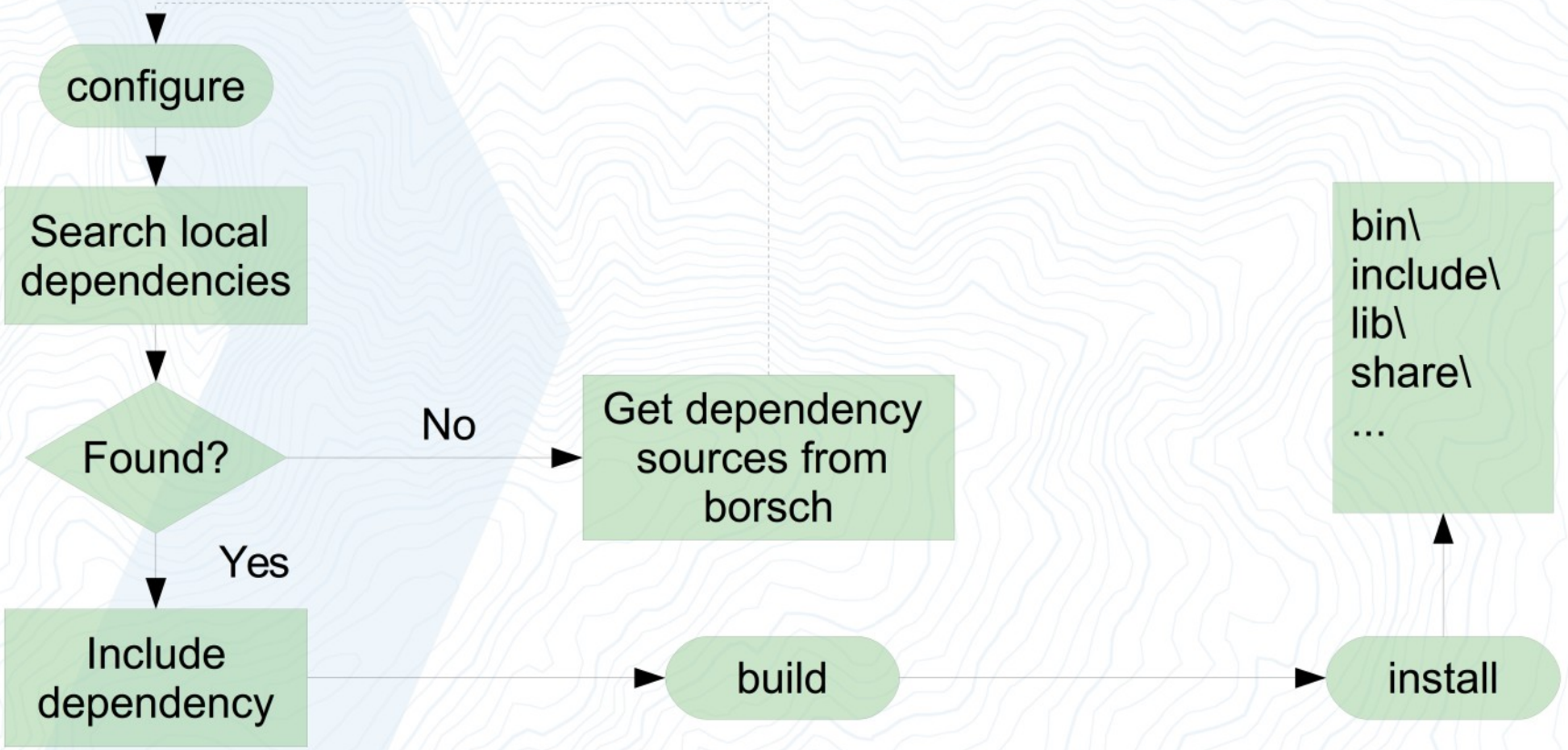
4. Link with libraries

   target_link_extlibraries(${NAME})

find_anyproject parameters

- Library (package) name
- VERSION and EXACT
- COMPONENTS, QUIET and MODULE
- DEFAULT
- REQUIRED
- SHARED
- CMAKE_ARGS

NextGIS Borsch logic



NextGIS Borsch: global variables

- EP_PREFIX
- EP_URL
- EP_BRANCH
- PULL_UPDATE_PERIOD
- PULL_TIMEOUT
- SKIP_GIT_PULL
- SUPPRESS_VERBOSE_OUTPUT

Library preparation steps

1. Library sources are build by CMake
2. Define install paths according to standard GNU paths - include(GNUInstallDirs)
3. Add the following export function:

export(TARGETS ${EXPORT_TARGETS} FILE ${EXPORT_NAME}-exports.cmake EXPORT_LINK_INTERFACE_LIBRARIES)

4. Dependencies must be included via find_anyproject function
5. Add latest scripts from Borsch repository to folder name cmake
6. Setup paths to modules

SET(CMAKE_MODULE_PATH ${CMAKE_SOURCE_DIR}/cmake ${CMAKE_MODULE_PATH})

7. Create configuration library file FindExtxxx.cmake with additional parameters

Current status

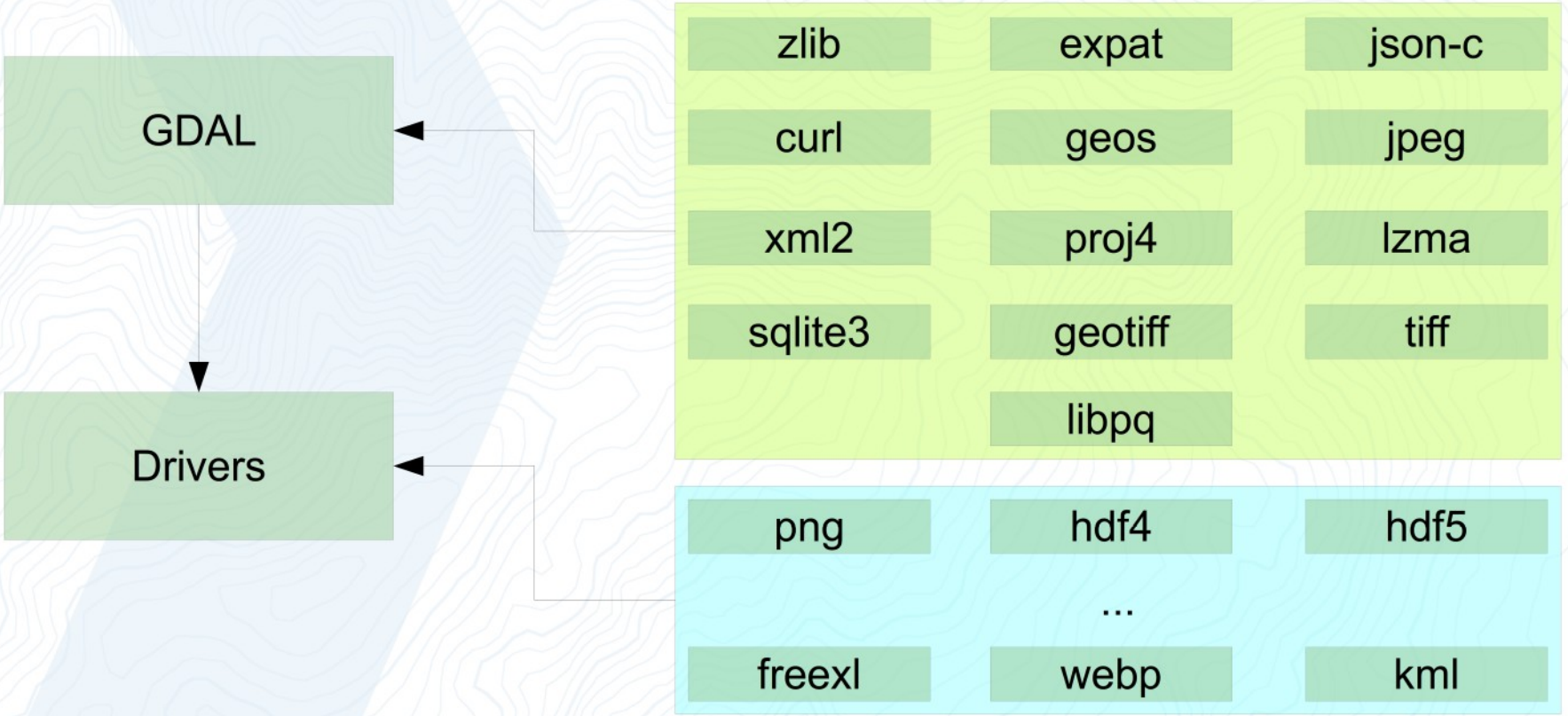
Currently more than **30** libraries are supported by Borsch:

- proj4
- geos
- gdal
- zlib
- json-c
- libpq
- tiff
- geotiff and etc.

We already build several products using libraries from Borsch:

- GDAL
- NextGIS desktop products (QGIS, Formbuilder and Manager)
- Mobile SDK for Android
- PostGIS

Borsch & GDAL



Borsch & PostGIS

Totally rewrited Mateusz Loskot CMake script

The borsch scripts were added (include(FindAnyProject)), than:

- **find_anyproject(ICONV DEFAULT ON)**
- **find_anyproject(PROJ4 VERSION 4.6 REQUIRED)**
- **find_anyproject(GEOS VERSION 3.4 REQUIRED)**
- **If raster support is required - find_anyproject(GDAL DEFAULT ON)**
- **find_anyproject(LibXml2 REQUIRED)**

Borsch & mobile SDK

1. Define using external libraries except zlib:

set(WITH_GDAL ON CACHE BOOL "GDAL on" FORCE)

set(WITH_GDAL_EXTERNAL ON CACHE BOOL "GDAL external on" FORCE)

set(WITH_SQLite3 ON CACHE BOOL "SQLite3 on" FORCE)

set(WITH_SQLite3_EXTERNAL ON CACHE BOOL "SQLite3 external on" FORCE) ...

2. Include external GDAL disabling unnecessary drivers.

find_anyproject(GDAL REQUIRED VERSION 2.0 SHARED OFF CMAKE_ARGS

-DENABLE_PLSCENES=OFF

-DENABLE_AAIGRID_GRASSASCIIGRID=OFF

-DENABLE_ADRG_SRP=OFF ...

Advantages & disadvantages

- simplicity
- crossplatform
- flexibility

- complicated dependencies resolution during building
- problem to support up to date scripts in all repositories
- necessary to have cmaked libraries with some additions
- common repository on Github for external libraries

Plans

1. conquer the world :)
2. library version checking in all cases
3. add more libraries
4. testing Borsch on more platforms
5. simplify updates for Borsch libraries

NEXTGIS
OPENSOURCE GEOSPATIAL SOLUTIONS

Questions?

info@nextgis.com www.nextgis.com @GisBishop