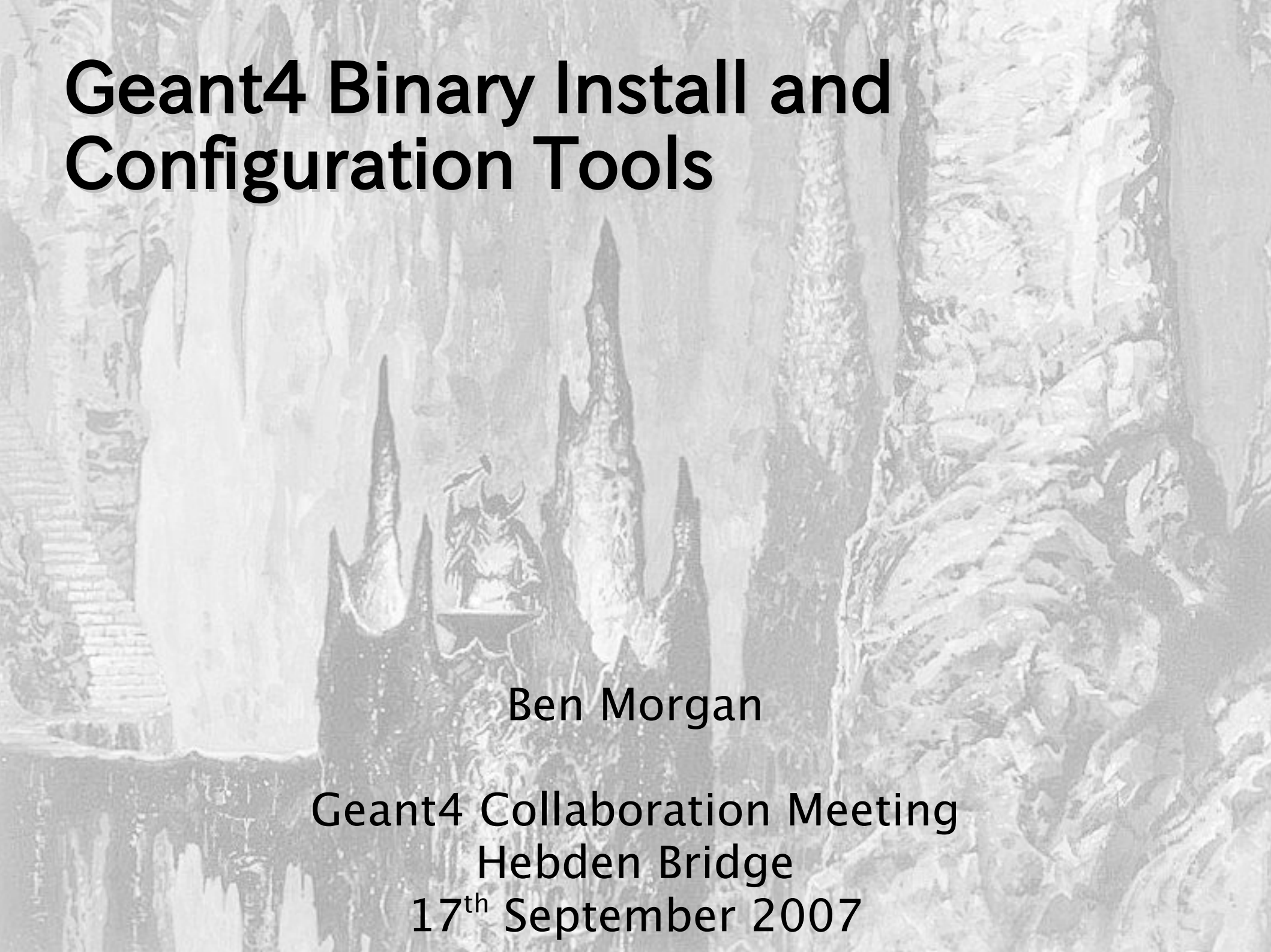

Geant4 Binary Install and Configuration Tools

Ben Morgan

Geant4 Collaboration Meeting
Hebden Bridge
17th September 2007

Binary Install Options

- Define binary install as:
  - A simple 'one click/command' way for users to install a precompiled version of Geant4.
- Simple idea, and leads to several issues
  - What packaging tools to use?
  - What to package?
  - Where to install files?
  - How to resolve dependencies on external packages?

System Packaging Tools

- Various flavours of OS supported.
  - Linux, Windows.
- Each with own packaging system(s)
  - Linux: rpm, deb (debs available)
  - Windows: msi, others?
  - Mac: dmg, fink (fink packages available).
- Emphasize that I have no experience in packaging for Windows or Mac.

- ***For now, concentrate on rpm on Linux issues.***

What to package?

- Global shared libraries only.
  - Is there a need for granular and/or static libs?
- Header files in single directory.
  - Needed for development.
  - Single directory provides best structure.
- Build system files for development.
  - All gmk files plus env.(c)sh scripts.
- Data files as separate packages.
  - Main package then has dependencies on versions.
- What about environments packages (g4py etc)?

Where to package?

- Should we follow the filesystem hierarchy?
  - Libraries in `/usr/lib` (`/usr/lib/geant4/Linux-g++`?)
  - Headers in `/usr/include/geant4`
  - Configuration files in `/usr/share/geant4/config`
  - Data files in `/usr/share/geant4/data/<name>`
- Alternatively, put everything under `/opt`:
  - `/opt/geant4/<version?>/lib`
  - etc...
- ***Later option useful if we want to support install of multiple versions.***

Dependencies

- Full Geant4 build and install depends on:
  - CLHEP(*), X11, OpenGL, Xm, Xaw, Inventor
- So a full binary install would require these on the user system.
- RPM is designed to handle that, but
  - Not all systems would have, or want, everything.
- One idea:
  - Try and package visualization drivers separately.
  - Packages `geant4`, `geant4-xm`, `geant4-xaw` etc.
  - May need to watch interlibrary dependencies...

Binary Install Planning

- A basic rpm for CLHEP is now available.
  - Needs final choice of location (/opt or /usr).
  - Final testing.
  - Geant4 plans for CLHEP?
- Sabar Salih at Manchester has an rpm for Geant4
  - Investigate use of this.
  - Use or adapt as neccesary
- Should discuss requirement for binary install further.

Configuration Tools

- Current system based on `metaconfig`.
- `Configure` script runs and sets up needed variables in shell scripts.
- Generated shell scripts used to run make:
  - `install.sh`
  - `move.sh`
- User interaction via 'question and answer' interface.
- Command line interface also available.
- ***Works well! But...***

Metaconfig Issues

- Uses a modular system for defining options (good)
  - Basic units raw shell scripts.
  - Just defining one option can require >200 lines of script.
- Metaconfig is an old system – very little documentation and few (if any) examples.
- Automatic configuration of include/library paths is a little awkward.
- If a test, e.g. for CLHEP, fails, will ask user for input rather than exit with failure.

A Better System?

- Primary requirements?
  - Ease of use for both users and developers?
  - Ease of maintenance/extendability?
- Distinguish configure and build processes.
- Gnu Autotools provides:
  - `autoconf` for configuration
  - `automake` for build
  - `libtool` for library builds
- ***One possibility – autoconf for configuration of existing Geant4 build system.***

Why Autoconf?

- Structure much like metaconfig
  - Top level `configure.ac` script.
  - If required, project specific set of m4 macros.

- What would you gain?
  - Huge amount of documentation/examples.
  - Simple coding for most configuration tests.
  - Easy to test external software functionality.
  - All options easily documented.
  - Easy variable export to scripts/Makefiles.
  - In principle better cross-platform support.
  - If required, easy integration of automake, libtool.

Planning for Configuration System

- Continue support for `metaconfig` system.
  - Bug fixes as needed.
  - Improve default options (easier user install).
  - Add critical extensions as needed.
- Continue development of `autoconf` front end?
  - Basic library options working.
  - Work on visualization options started.
  - Estimate 3-6months work required to get to beta.

Other Options

- Scons is a relatively recent python based build system.
- Doesn't yet implement configuration tasks.
- Also, whole build, not just configuration, would have to use it.
- Nevertheless, something to keep an eye on.

Other Configuration Ideas

- A few (personal) speculative and much longer term ideas.

- Classical autoconf examples generate 'config.h'
  - `#define` s variables for optional parts of software.
  - Might be cleaner than environment variables (-D)?

- Use '.geant4rc' file to control runtime options.
  - Could be read and parsed by application/kernel.
  - Might be an easier point of contact for core and system than environment variables?
  - Can also have per process files.