Ideas on Software Distribution Model

Librarians and Integrators Meeting (4.6.2013)

Benedikt Hegner, Pere Mato

New CMake build infrastructure

Fairly simple solution for building exterals now

- get or setup cmake
- checkout lcgcmake package from SVN
- setup C/C++/Fortran compilers
- create workspace area
- configure with **cmake**
- build with make

- On Ixplus set PATH to use one of latest CMake versions (default is 2.6) export PATH=/afs/cern.ch/sw/lcg/external/CMake/2.8.9/Linuxi386/bin:\${PATH}
- Checkout the lcgcmake package from lcgsoft SVN repository svn co svn+ssh://svn.cern.ch/reps/lcgsoft/trunk/lcgcmake
- Create a workspace area in which to perform the builds mkdir lcgcmake-build cd lcgcmake-build
- You may need at this moment to define the compiler to use if different from the native compiler

source /afs/cern.ch/sw/lcg/external/gcc/version/platform/setup.(c)sh

- 5. Configure the build of all externals with cmake cmake -DCMAKE_INSTALL_PREFIX=../lcgcmake-install ../lcgcmake
- In order to build against the existing external repository use the option -DLCG_INSTALL_PREFIX=/afs/cern.ch/sw/lcg/external to tell the system to look for packages in the LCG area.
- Build and install all external packages make -j
- Or to build a single external package make -j <package> (use make helpto see the list of all available packages)
- You may need to restart de build of a package from beginning in case of obscure errors. The best is to clean a specific package

make clean-<package>

http://sftweb.cern.ch/spi/HowtoBuildWithCMake

New CMake infrastructure

- New release build infrastructure tested every night
 - Results shown on cdash.cern.ch
- Outstanding problems:
 - CORAL and COOL aren't integrated yet
 - The nightly summary page isn't integrated here

Login All Dashboards	Calendar	LCGSoft Previous	Current	Pro	oject						8	Sunday, I	May 05 2	2013 21	:34:08 CE	est 🔊
file changed as of Sun xperimental	day, May 05 20	013 - 03:00 CEST									Show F	ilters Si	imple Vie	ew A	uto-refre:	sh Help
					Update		Configure			Build			Test			
Site		Build Name	e	Files	Time	Error	Warn	Time	Error	Warn	Time	Not Run	Fail	Pass	Time	Build Time
acphsft20	🗯 x86	_64-mac108-gcc4	42-opt 🔍	0	0s	0	0	30s	50	1	1h 28m					1 hour
gapp07.cern.ch	∆ x86.	_64-slc6-gcc48-o	opt @	0	0s	0	o	12s	4	1	1h 20m 42s				~ _	t moi
build175.cem.ch	∆ x86.	_64-slc5-gcc43-o	pt	0	6 s	0	0	12s	0	0	1h 9m 42s				to	build
			0.40								460				á	and g

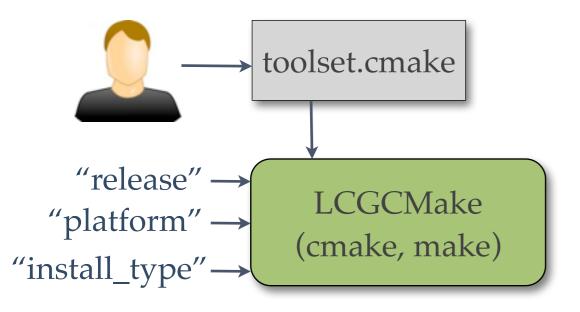
Building Full Releases

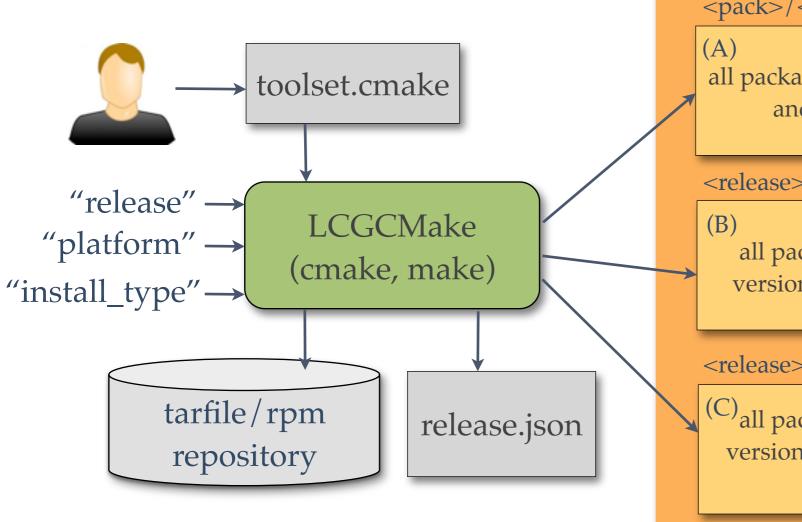
- We propose a **new way** of providing releases
- Preparing the full stack is a much cheaper operation now

two tiets

- Until a real release has been made, new build products will go into an experimental area only
 - the gcc48 build already follows this approach
 - Would like to apply the same to *all platforms* in the -preview slot soon!
- Once the release is cut, it will be made available in
 - CVMFS (for sure)
 - AFS (is that still required?)

• But how will such a release look like?





AFS/CVM-FS <pack>/<vers>_<id>/<platf> all packages with all versions and all platforms <release>/<pack>/<ver>/<platf>

> all packages with single version and all platforms

<release>/<platf>

(C)_{all packages with single} version compacted and all platforms

