Agenda

The focus topic is packaging. Others will be working on additional projects as well. See the Discourse link for more information. Non-Kitwareans are welcome to attend one of our offices (KHQ-Clifton Park); (KRS-Chapel Hill); (KSW-Santa Fe); and (KEU-Lyon, France) – but please let us know you are coming so we can order you lunch. Or join via Google Hangout.

9:00 Master of Ceremonies Dave DeMarle kicks it off
(talking about goals & process; focus on packaging)

Project Work

11:30 Dave leads a discussion on VTK Next
(future directions for VTK)

Working Lunch - lunch provided

Project Work

Room is available until 4:00pm.

Attendees:
KHQ
- Sankhesh
- D DeMarle
- Will
- Aylward
- Berk
- TJ
- Kyle
- Ben
- Sean McBride
- Utkarsh
- Lipa

KRS
- Ken
- Cory
- JC
- Thompson

McCormick
Forrest
Hastings

KEU
Joachim
Michael
Nicolas
Charles
Max

KSW
Sebastien

Previous VTK Hackathons:

1. Feb 2013 - “dashathon”
2. Feb 2014 - “coveragathon”
3. Oct 2014 - “bugathon”
4. Jul 2016 - “bugathon 2”
5. April 2017 - “examplethon”
6. Sept 2017 - “depracathalon”
7. Oct 2018 - “mergathon”
8. April 2019 - “mergathon 2”

Today Nov 2019 - ~~“packathon”~~ yackathon more like it

You are free to work on any topic you like. For minor items see the issue tracker, for bigger items see the discussion list for this afternoon.

Please do partake in the mindshare via google hangout and discourse. As usual, the hope is that if we get enough people focussed together we can make significant inroads on something important.

VTK Contribution guide: https://gitlab.kitware.com/vtk/vtk/blob/master/CONTRIBUTING.md

Ideally we will end up today with a new VTK package somewhere, tested no less than nightly and submitting to VTK dashboards.

Back in the day, part of VTK’s success was that it was comparatively easy to get started with.

https://vtk.org/Wiki/VTK_Get_The_Source_Code_Via_WinCVS
Today VTK is just as easy to get started with! Standards have improved and people expect a lot more today.

Kitware's focus is primarily developing new features for customers, many of whom use VTK only indirectly.

Users (especially new users who are considering VTK) don't want to get bogged down with technical issues of the code construction. They need to quickly evaluate what it can do for them.

Fortunately volunteer packagers have picked up much of the slack.

- debian - link please
- redhat - https://koji.fedoraproject.org/koji/buildinfo?buildID=1404601
- homebrew - https://github.com/Homebrew/homebrew-core/blob/master/Formula/vtk.rb
- macports
- fink
- cygwin
- mingw - https://github.com/msys2/MINGW-packages/commits/master/mingw-w64-vtk
- Maven - https://gitlab.kitware.com/vtk/vtk/blob/master/Wrapping/Java/README.txt
- PyPi
- Conda - link please
- Windows SDK (have to search old mailing list)

The problem:
there are a lot of platforms (OS & Language) ***x***
there are a lot of options when you configure VTK ***x***
there are a lot of dependencies ***x***
there are a lot of VTK versions.

Ideally it would be trivial for a new user to get up and running with the latest release on any popular platform. Ideally it would also be trivial for dependent applications to import VTK.

Questions to be answered:

1. What specifically do we need to do to make packaging straightforward?
2. How do we extend our regression testing infrastructure to make sure it stays working?
3. What platforms do we need to support better and how is the package management system structured there?

Experiences from Packaging People:

DeMarle - vtkPython.exe (standalone vtk python interpreter)

*** when was/is the vtk packaging project in active development?**
VTK 7.x (2016)

*** where is the repository/branch and top level doc?**
https://gitlab.kitware.com/vtk/vtk-superbuild/commits/master - a fork of ParaView's superbuild
*** where is/was the package made?**
one linux, one mac, and one windows box in Dave's office
*** where does the produced package get pushed to/pulled from?**
vtk.org downloads page
*** where is/was the CI?**
none - it was manually run only during release candidate cycle
*** what is the current state of the project?**
dead - It didn't have sufficient CI to sustain it and it didn't cooperate with system python.

Sebastien - Maven (Java)

* when was/is the vtk packaging project in active development?
- VTK 7.x (2016)

* where is the repository/branch and top level doc?
- https://gitlab.kitware.com/vtk/vtk/blob/master/Wrapping/Java/README.txt

* where is/was the package made?
On Linux, Windows in Ben's office
On Mac in Seb's office (old mac-mini)
* where does the produced package get pushed to/pulled from?
sent to customers via Seb's Google Drive
* where is/was the CI?
No
* what is the current state of the project?
Same as the vtk-superbuild state

Aashish - Spack ...

no VTK package, only ParaView

* when was/is the vtk packaging project in active development?
* where is the repository/branch and top level doc?
* where is/was the package made?
* where does the produced package get pushed to/pulled from?
* where is/was the CI?
* what is the current state of the project?

JC/Ben - PyPi

* when was/is the vtk packaging project in active development?
8.0, 8.1, ~~8.2~~
update for VTK master/9.0
started 2 years ago
* where is the repository/branch and top level doc?
was -> https://github.com/KitwareMedical/VTKPythonPackage
soon vtk itself can make a wheel, infrastructure to run it will be very simple
https://gitlab.kitware.com/vtk/vtk/merge_requests/5543

* where is/was the package made?
to set up a linux, mac and windows nightly submitter
* where does the produced package get pushed to/pulled from?
Ultimately packages will be published on PyPi
Development (aka Nightly) wheel will be published daily either on GitLab or https://vtk.org/download/
* where is/was the CI?
it will be VTK buildbot
* what is the current state of the project?
active development, a requirement for VTK 9.0.0.rc1

Dan Lipsa - Conda

- external packagers made conda forge only for releases (Matthias Belcher v 7.0): https://github.com/conda-forge/vtk-feedstock
- now, Kitware for LLNL Climate via Kitware (VCDAT): https://github.com/CDAT/conda-recipes/tree/master/vtk-cdat Besides vtk-feedstock, vtk-cdat does cross-compiling on linux (uses centos 6) and mac os (uses an old version of mac). We should merge vtk-cdat into vtk-feedstock.
- see Conda Press project for converting conda package into pypi.

Kyle - Debian

* when was/is the vtk packaging project in active development?
last year VTK 8 https://salsa.debian.org/science-team/vtk8 - has not yet been accepted into Debian
* where is the repository/branch and top level doc?
was manual at debian
Kyle's project wrote routines that translate cpack package to a debian package
* where is/was the package made?
https://gitlab.kitware.com/debian/dh-cmake
https://gitlab.kitware.com/debian/vtk
* where does the produced package get pushed to/pulled from? https://apt.kitware.com/
debian science team's input - cpack's packages not sufficient thus Kyles's approach, need to be more complete/consistent with copyright
* where is/was the CI?
Dashboard submitter, but it died
* what is the current state of the project?
close but not finished, needs updated for VTK 9's cmake, needs dashboard, dh-cmake needs to be completed (docs, acceptance into Debian) NEEDS FUNDING

Orion - Fedora

* when was/is the vtk packaging project in active development? Current (2019), v8.2.0, 8.1.1
At least since 2007 - vtk 5
* where is the repository/branch and top level doc? https://src.fedoraproject.org/rpms/vtk
* where is/was the package made? on the Koji build system (RedHat/Fedora)

* where does the produced package get pushed to/pulled from? See https://fedoraproject.org/wiki/Koji for a description of Koji
* where is/was the CI? https://koschei.fedoraproject.org/package/vtk
* what is the current state of the project? Active, only 1 issue being tracked (nVidia) https://bugzilla.redhat.com/buglist.cgi?quicksearch=%3Avtk

Matt McCormick - WebAssembly

* when was/is the vtk packaging project in active development?
  VTK WebAssembly builds have been available on DockerHub since 2019-02-05
* where is the repository/branch and top level doc?
  https://github.com/thewtex/VTK/tree/emscripten-updates (these patches could be pushed upstream)
* where is/was the package made?
  https://hub.docker.com/r/kitware/itk-js-vtk
* where does the produced package get pushed to/pulled from? https://github.com/InsightSoftwareConsortium/itk-js/tree/cdc52a35c2f958bd3ff5e1ead865e78b5eacc4cd/src/Docker/itk-js-vtk
* where is/was the CI?
  https://circleci.com/gh/InsightSoftwareConsortium/itk-js
* what is the current state of the project?
  Done
  - VTK (minus rendering) compiles to WebAssembly / Emscripten
  - VTK processing pipelines can be created and executed via itk.js
  - Geometry file formats can be read into vtk.js PolyData for rendering
    - Want to learn VTK / WebAssembly and contribute to the community? -- pick a module and add support for it
  - Next
    - Create a "lean" build that results in smaller binaries
    - Add rendering support
    - Add a vtkJSONDataSetReader
    - Incorporate into vtk.js
    - Incorporate more Emscripten patches
    - Add / test multi-threading

Let's hack, and make more notes here ...

Smaller components (a vtk core package), internal and external components would be ideal
how to structure code/projects?
how to structure cmake for the external components so they can be built independently
a technical problem with dependency's versions and exposed VTK symbols in them ex mpi vs parallel mpi
findpackage and components behavior, what will consumers and how to use it?

The fact that ParaView doesn't support an external VTK is not helpful. Same for infrequent VTK releases.

opinions -> we should make vtk easier to package for all, set up nightly regression testing for one or two of the most important

9.0 will not release without a wheel

+ need to figure out what modules/features go under what name (we probably won't ship wheel w/ MPI enabled, but allow others to build a wheel w/ MPI... we need to find a naming scheme that differentiates them)
+ buildbot is a big unknown

Conclusion

JUICY ITEMS TO WORK ON TODAY:

- license formatter/verifier
    - Instructions for writing a formatter/verifier is here
- PyPi finish
    - Rebase current branch against latest master: Jc - **Done** ✅
    - Test current branch on
        - Windows: Jc
            - Python 3.6:
            - Python 3.7:
                - Build the wheel fails with this error - **Solved**
                - Problem was that cmake was not in the path and that CMAKE_EXE env. Variable had to be set. Will improve setup.py to report an informative error message.
                - ~~WrappingPythonCore~~ vtkCommonCorePython target **fails to link against pythonXY.lib**
                - This is due to auto linking happening in pyconfig.h. See here
                - An issue has been reported upstream. See https://bugs.python.org/issue38728
                - To mitigate this, I suggest we **ensure the link directory is propagated.**
            - Python 3.8:
                - previous vtk build error was due to use of 3.8.0a2 on the build machine, switching to 3.8.0 addressed it - **Solved**

            - Building the wheel fails with this error - **Solved**
        - macOS : Dave
            - Python 3.6:
            - Python 3.7: **Done** ✅
            - Python 3.8:
        - Linux: Ben
            - I suggest to use the manylinux docker images
    - Update buildbot to build VTK against python 3.6, 3.7 and 3.8: Dave/Ben
        - Ben notes available here
        - For each MR:
            - What is overhead ? Do we want this ? Or may be only 3.8 ?
                - overhead of turning VTK_WHEEL_BUILD on time seems to be minimal on my mac laptop build
            - Adding a +nightly buildbot feature makes it not run for plain `Do: test` (`Do: test --nightly` is available)
        - https://kwgitlab.kitware.com/buildbot/buildbot/merge_requests/1753
    - Add test to trigger wheel building: Jc ?

11:30

VTK Next
discussion list
. reference to vtk coding guidelines:
https://docs.google.com/document/d/1nzinw-dR5JQRNi_gb8qwLL5PnkGMK2FETlQGLr10tZw/edit

2:10

SmackDown - vtkSMP vs vtk-m, how long do we keep dividing developer attention?

- What is performance on current hardware now, what will it be in the future given hardware trends? Rumor has it that vtk-m is not as performant on CPUs now which are far more prevalent than high end GPUs. Rumor also has it that the learning curve is much worse too. ParaView binary has plugins for both and a timerlog, thus an easy way to get rough comparisons to get a feel.
- No question we have to support GPUs well for the future. The question is, at what point does it not make sense to focus any more resources on vtkSMP and write code that will have to be rewritten?
- even more outreach is needed, more vtk-m competent developers are also needed
- idea: an example that adds optional vtk-m core to a core/filter to accelerate adoption
- idea: encourage paraview developers to prefer vtk-m
- idea: can we write a simple parallel_for for vtk-m - no