Alternative Implementations of “Late Hydration” or “ipex” or “performantly redeploying PEX files by sharding their requirements”

Comments Appreciated!

Problem

Redeploying pex files full of many extremely large 3rdparty requirements (tensorflow, etc) into our datacenter currently takes a very long time, since we upload them all at once into an internal artifact store, and then pull down the entire pex file (>1GB) before executing it. This slowness to redeploy then also affects multiple of our internal python development workflows and tooling for machine learning (including a Jupyter wrapper developed by @kwlzn) which depend on executing a pex file within the datacenter -- in that case, modifying any python source files in our monorepo currently requires waiting several minutes for changes to be usable within that Jupyter notebook.

As far as we are aware, other users of pex who package large machine learning applications also suffer from this issue and do not have an easy workaround.

Solutions Under Consideration

A useful way to look at approaching the problem is on two sides: figuring out how to ship pex files without all the downloaded deps which avoid paying a huge upload/download cost at build time, and figuring out how to then get the right deps into the pex before it tries to run.

Solution (Part 1): “Dehydrated” or “ipex”: pex files without deps embedded in the pex

We would like to be able to ship around "dehydrated" pex files without 3rdparty requirements embedded in the pex, and resolve ("hydrate") them before executing the pex. This removes one half of the current process of synchronously waiting to upload and download 3rdparty requirements, and moves the remaining download part off the critical path of the entire redeploy process.

This is half of what is implemented in https://github.com/pantsbuild/pex/pull/787 -- if the `--dehydrated` flag is provided, pex will then avoid putting any resolved requirements in the pex file, instead adding a `dehydrated_requirements` field to PEX-INFO with the exact versions of all transitive requirements.

Because the requirements to hydrate were already resolved when building the pex, we know all the exact versions of all the transitive dependencies to resolve before running the pex. With an extension of this, we can also avoid having to pull down large wheels onto the developer's laptop, by doing some form of metadata-only resolve (also proposed by @kwlzn). *This extension has not been implemented yet.*

While the method of doing "late hydration" can be done by another tool, it's not clear that we can avoid the 3 phases of:

- *Downloading massive requirements to the developer's laptop.*
- *Uploading a massive pex file to our artifact distribution utility.*
- *Pulling down the massive pex file into the deployment environment.*

without changes to pex such as the proposed `--dehydrated` flag in #787.

Using constraints files with transitive requirements may be able to effectively address this requirement!

Solution (Part 2): "Late Hydration": resolving requirements just before runtime

We would like to be able to run "dehydrated" or "ipex" pex files unmodified, for the usual reasons: because then we can avoid modifying the rest of our deployment workflow to do this, and the only required change would then be to just pipe in an option to create dehydrated pex files into pants for us to implement this.

One particular implementation of this which should *not* be used directly is in #787, which will resolve all dehydrated requirements every time the pex starts up. A modification proposed by @kwlzn is to treat it like the `zip_safe` attribute, and resolve the appropriate requirements the first time the pex is run. Note that the proposed `--dehydrated` flag would resolve all transitive requirements up front, so the resolve can be performed with `transitive=False`.

The current implementation in #787 introduces a new environment var named PEX_BOOTSTRAP_HYDRATION_INDICES which provides pypi-like indices to resolve against at bootstrap time. This constrains the initial "late hydration" process to only specifying which indices to resolve against, compared to the breadth of resolver options normally available at build time.

Alternative "Hydration" Solutions

These are alternative solutions we're considering for providing the "dehydrated" requirements when the pex is first run.

Requirement(s)

The implementation in #787 was a first draft of this functionality, and there was quite reasonable concern about introducing runtime requirement resolution to pex itself. For alternatives, there is at least one capability we would want to maintain compared to the pex-only solution:

1. We would like to make the resulting pex “self-contained”, i.e. runnable without a separate deploy script. By definition, executing a “dehydrated” pex would still require hitting the network at runtime, but only to hit a pypi-compatible index to resolve requirements.

virtualenv

As @illicitonion proposed previously and as @jsirois has proposed in #789, virtualenv can be used to provide the appropriate resolved requirements, especially once we align our resolver with pip’s. In particular, this comment:

I'm still stuck back on why PEX is the right tool for the job at all here. It sounds like your requirements / acceptable actions include:

1. *It's ok to resolve pinned artifacts on the target host at PEX boot time.*
2. *You want to resolve the minimal set of things.*
3. *You want an isolated virtual environment.*

A virtualenv + a pinned requirements file with hashes and a pip install --no-deps --only-binary :all: --find-links=<internal flat repo> --index-url=<internal pypi> --requirement requirements.locked.txt does exactly what you want. Is it perhaps the case that its only because Pants does not support this that you want to jam this functionality into pex?

To put a finer point on the last - with #781 in flight, the pex resolver will == the pip resolver. When Pants upgrades to pex 2.0.0 with the pip resolver, would pants generating a lockfile with hashes be enough here? You deploy the lockfile and run pip install .. which will do the minimal update.

Open Questions

- Is there a way to produce “dehydrated” pex files without pex intervention?
    - We may be able to make use of the alternate effort to support requirement constraints files to get the transitive resolve we want for the “dehydrated” part.