Transitive static libraries

Please read Bazel Code of Conduct before commenting.

- **Authors**: fmeum
- **Status**: Draft | In review | **Approved** | Rejected | In progress | Implemented
- **Reviewers**: oquenchil (LGTM)
- **Created**: 2023-01-27
- **Updated**: 2023-03-29
- **Discussion thread**: #1920

Overview

The existing C/C++ rules can produce executables (`cc_binary`) and shared libraries (`cc_binary` with `linkshared = True` or the currently experimental `cc_shared_library`) as final, deployable build outputs. These rules collect intermediate build artifacts from all their transitive dependencies' `CcInfo` providers.

Bazel currently doesn't offer a way to produce a single static library (also called an "archive") from a transitive collection of intermediate targets. Since open-source library projects (such as Tensorflow) often want to make static libraries available as release artifacts, this limitation should be lifted via a canonical, officially supported rule for this purpose.

Proposal

Add a cc_static_library rule to Bazel, implemented as a Starlark built-in rule. The rule is backed by a new action name and a new well-known toolchain feature, which makes the actual command line of the action producing the static library configurable via toolchains.

Toolchain changes

Action & feature

`validate-static-library`: A new well-known name for an action that validates a given static library, e.g. by detecting duplicate symbols. Action invocations are always of the form:

```
<tool_path> <static library> <validation output file>
```

`static_library_validation`: A new feature that controls whether `cc_static_library` registers `validate-static-library` as a validation action.

Default toolchain implementations

As static libraries can contain duplicate symbols (the archiver does not check for them), the validation action should at least ensure that there are no duplicates. The auto-configured toolchains can provide the following implementations, depending on the available `"nm"` tool:

nm

Duplicate symbols can be checked for with a simple shell script:

```
#!/usr/bin/env sh
set -eu

DUPLICATE_SYMBOLS=$(
  "%{nm}" --demangle --print-file-name "$1" |
  LC_ALL=C sort --key=3 |
  uniq -D --skip-field=2)
if [[ -n "$DUPLICATE_SYMBOLS" ]]; then
  echo "Duplicate symbols found in $1:" | tee "$2"
  echo "$DUPLICATE_SYMBOLS" | tee --append "$2"
  exit 1
else
  touch "$2"
fi
```

Rule definition

Name

`cc_static_library`

Attributes

- `data`: same behavior as on `cc_library`
- `deps`: label list with entries required to provide CcInfo

Providers

- `DefaultInfo`: the merged archive

- `OutputGroupInfo`
    - `linkopts`: A text file containing the deduplicated transitive linkopts, with one flag per line. Consumers of the transitive static library may need to add these flags to their own linker invocation.
    - `linkdeps`: A text file containing one line per transitive dependency that provides a shared or static library but not object files, in the form "`<exec path> <owner label>`". Consumers of the transitive static library may need to add these artifacts to their own linker invocation.
    - `targets`: A text file containing the targets that contributed an object file, with one label per line.
    - `_validation`: The output of the `validate-static-library` action, if the feature `static_library_validation` is enabled.

The rule does **not** provide `CcInfo` as it is not meant to be consumed within the build that produces it. If users need to do this and are aware of the risk of ODR violations this introduces, they can wrap the output of the rule in a `cc_import`.

Rule implementation considerations

- The rule is gated behind a new `--experimental_cc_static_library` flag and implemented as a Starlark builtin rule with a Java shim.
- `ctx.actions.args` is used for the input files (for memory efficiency) while `get_memory_inefficient_command_line` is used to get the flags and output file path (for toolchain configurability).
- For each transitive `LibraryToLink`, all `pic_objects` (or `objects` as a fallback) are merged into the static library output.
- Param files are used if the existing archive_param_file feature is enabled.
- The input files to the archiver action can only be obtained by flattening the depset of transitive linker inputs. This could be prevented by introducing a variant of depset that includes a "filterMap" style callback.
- `linkopts`, `linkdeps` and `debug_targets` are collected at execution time using `ctx.actions.expand_template`'s `computed_substitution` parameter. For the `linkdeps` output group artifact, `static_library` is preferred over `dynamic_library`.

Compatibility

Adding new rules, actions and features does not affect backwards compatibility.

Document History

Date	Description
2023-01-30	First proposal
2023-03-26	Updated to address first round of comments
2023-03-29	Updated to address second round of comments