OpenMP offload from static libraries

foo.c

```
void foo() {
#pragma omp target
 printf("Target is executed on the \"%s\".\n",
    omp_is_initial_device() ? "host" : "device");
}
```

bar.c

```
extern void foo();
int main() {
  foo();
  return 0;
}
```

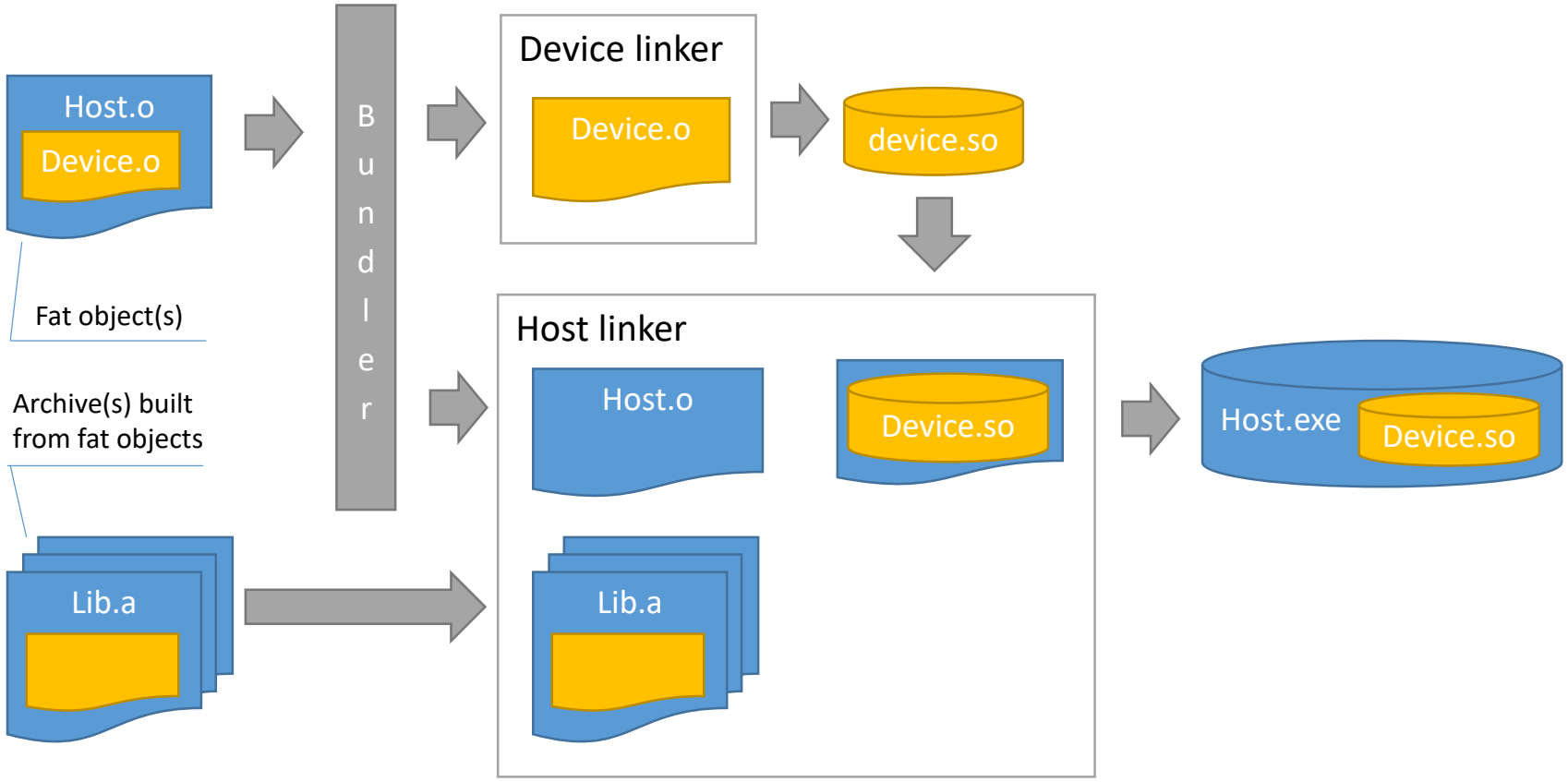
```
$ clang -fopenmp -fopenmp-targets=x86_64-pc-linux-gnu -c foo.c
$ ar crv libfoo.a foo.o
a - foo.o
$ clang -fopenmp -fopenmp-targets=x86_64-pc-linux-gnu bar.c -L. -lfoo
$ ./a.out
Target is executed on the "host".
```

vs.

```
$ clang -fopenmp -fopenmp-targets=x86_64-pc-linux-gnu foo.c bar.c
$ ./a.out
Target is executed on the "device".
```

OpenMP offload from static libraries is not supported, but it is a highly desired feature!

Current OpenMP offload linking flow



Unbundling is done only for object files, device code from static libraries is ignored.

Suggested change to OpenMP offload linking

Add a new intermediate link step which will do partial linking of object files and static libraries provided in command line

- All required objects (dependencies) will be pulled in from static libraries
- Partial linking will produce a single fat object file with device parts concatenated in corresponding ELF sections

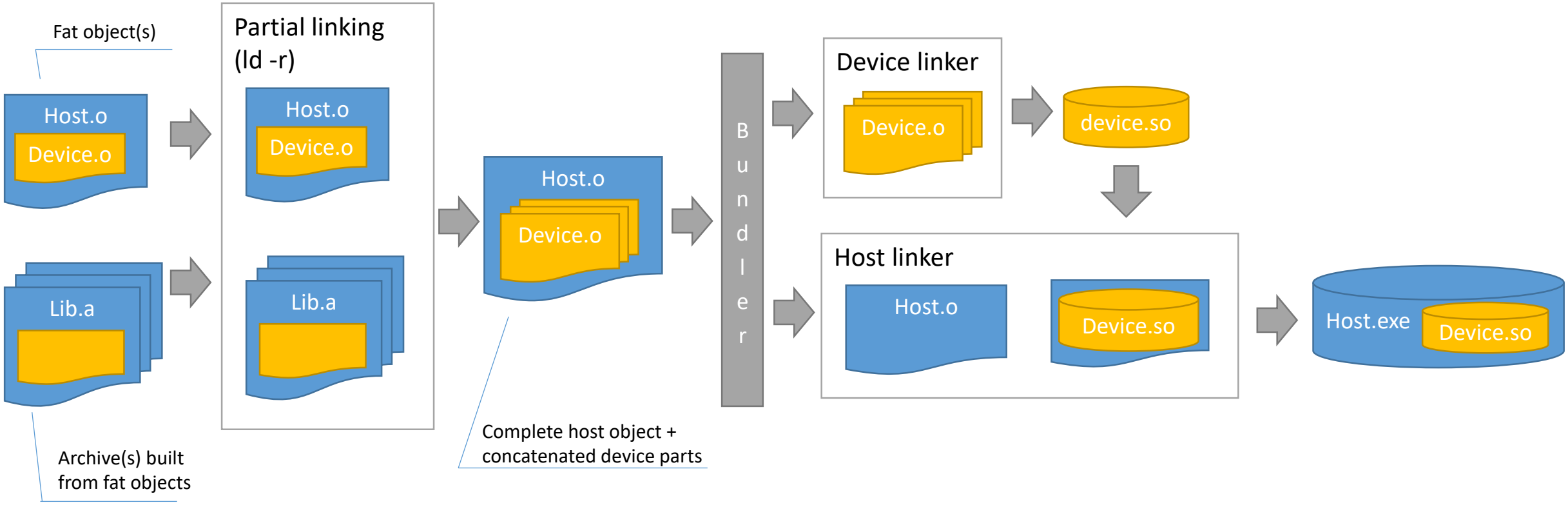
Extract device parts from the partially linked fat object

- Expect to get one or more object files for each offload target
- Offload bundler has to be enhanced to extract multiple concatenated device objects from the given fat object

And continue linking as it is done now

- Link device binary for each offload target from the extracted device objects
- Create an offload linker script for embedding device images into the host program
- Link host binary using the linker script and the host part of the partially linked object

Suggested change to OpenMP offload linking



Add partial linking for fat objects and libraries. It will pull all required fat objects from static libraries.