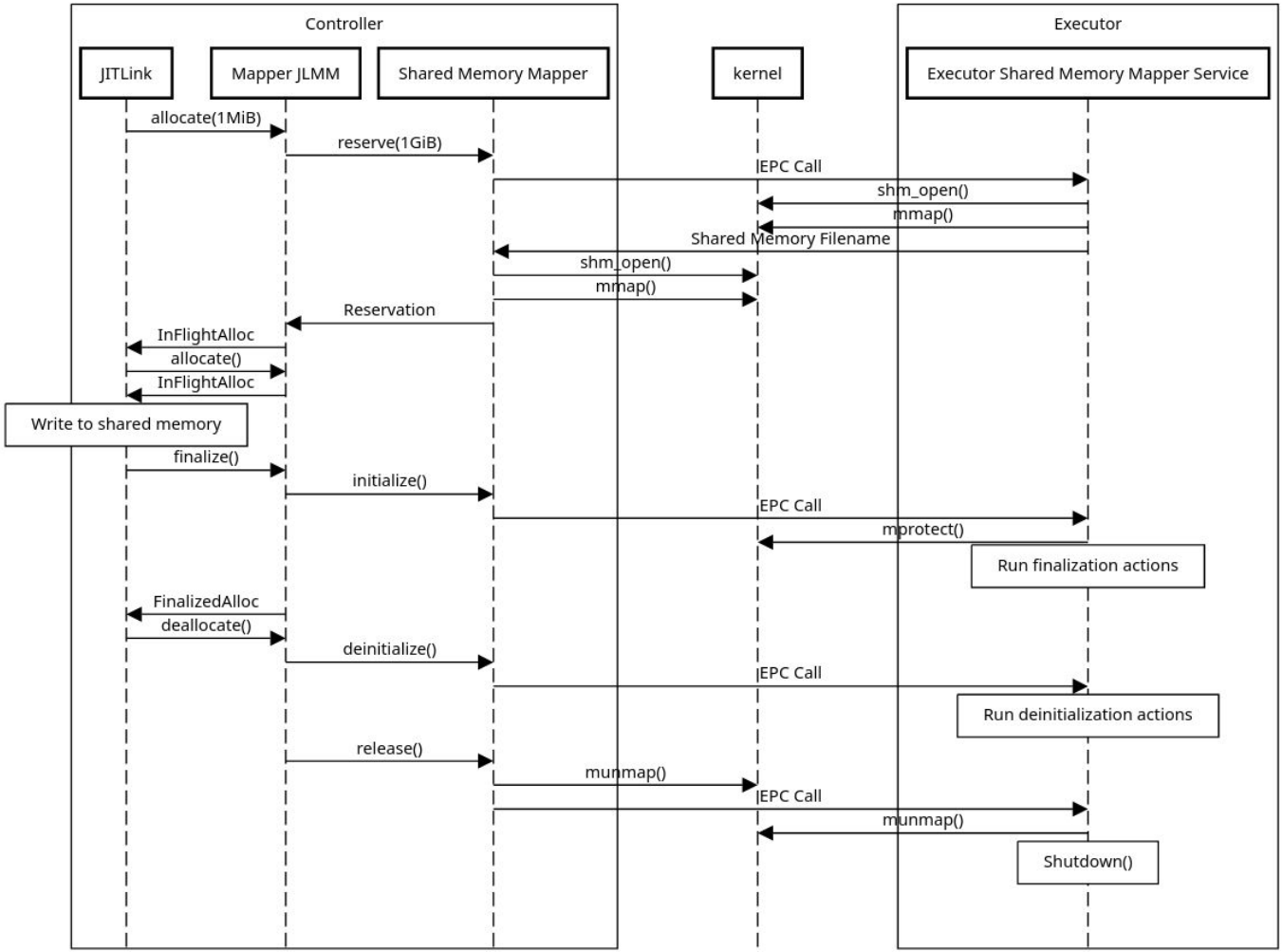
Shared Memory Based JITLink Memory Manager

Student: Anubhab Ghosh

Mentors: Vassil Vassilev, Lang Hames, Stefan Gränitz

Mapper JITLink Memory Manager

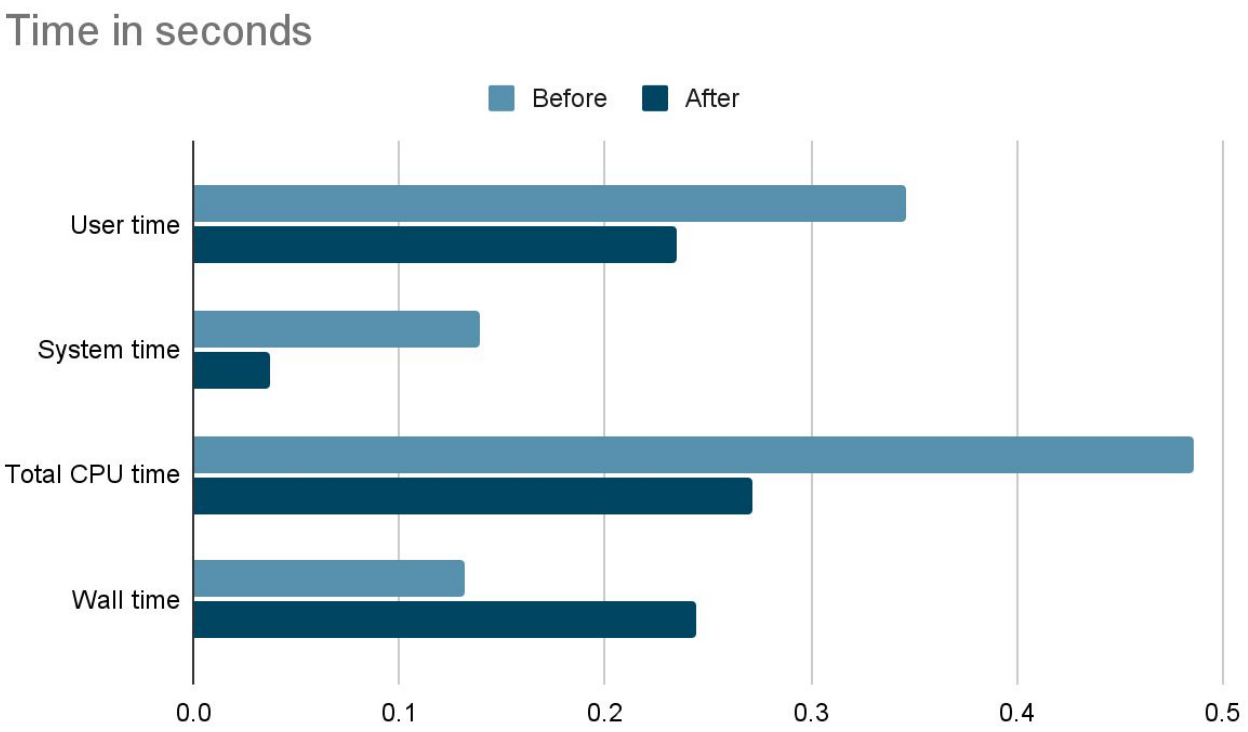
- It takes a MemoryMapper and uses it for all low level operations.
- It reserves a large chunk of memory on first allocate().
  - By default multiple of 1MiB on Windows and 1GiB everywhere else.
- It uses a slab allocator to allocate memory.
  - `llvm::IntervalMap` is to keep track of free memory regions.
    - It is possible to reuse freed memory.
    - It can perform automatic coalescing of memory regions.


Controller
Executor
JITLink
Mapper JLMM
Shared Memory Mapper
kernel
Executor Shared Memory Mapper Service
allocate(1MiB)
reserve(1GiB)
EPC Call
shm_open()
mmap()
Shared Memory Filename
shm_open()
mmap()
Reservation
InFlightAlloc
allocate()
InFlightAlloc
Write to shared memory
finalize()
initialize()
EPC Call
mprotect()
Run finalization actions
FinalizedAlloc
deallocate()
deinitialize()
EPC Call
Run deinitialization actions
release()
munmap()
EPC Call
munmap()
Shutdown()

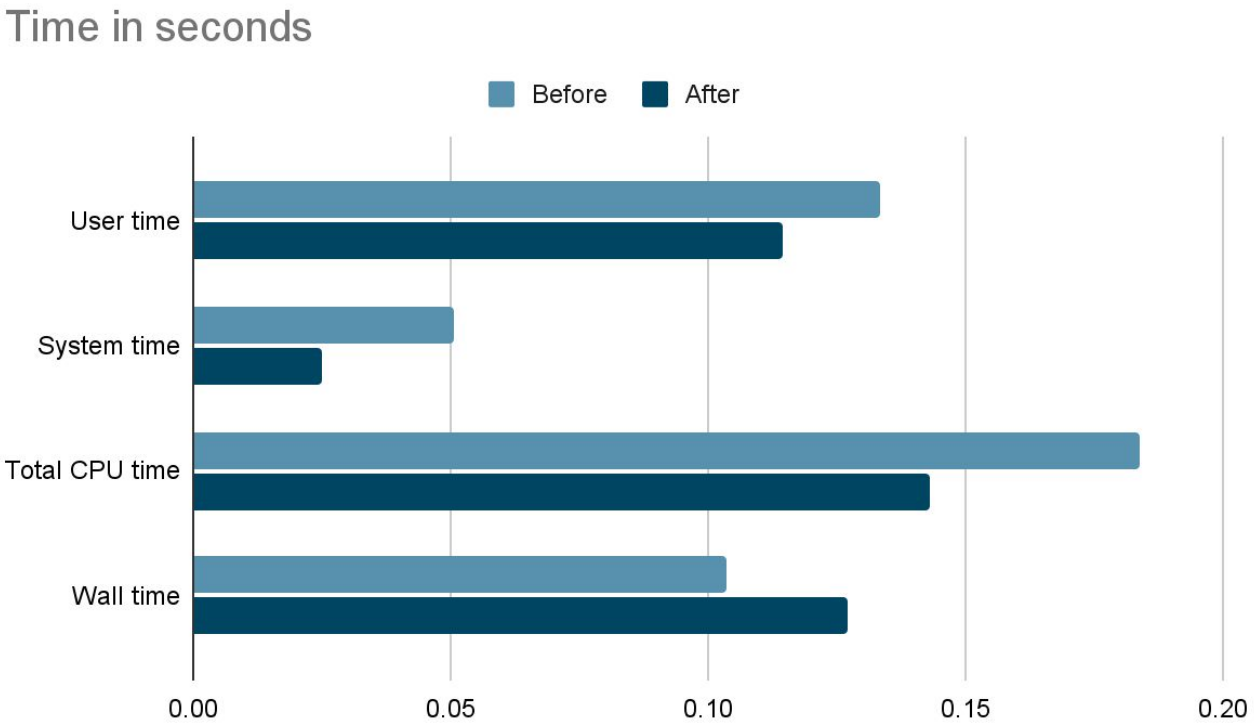
Current Progress

- `MemoryMapper` interface
  - `InProcessMemoryMapper` using sys::Memory APIs
  - `SharedMemoryMapper` using POSIX and win32
- `MapperJITLinkMemoryManager` implementation
  - It can use one of the above memory mappers
  - It has a slab allocator.
- `llvm-jitlink` tool integration
  - `InProcessMemoryMapper` is enabled by default now
  - Shared memory can be enabled with `--use-shared-memory` switch when running with `--oop-executor=` or `--oop-executor-connect=`
  - It can run projects that normally run with llvm-jitlink. The C-Ray raytracer and Python interpreter seems to work

CPython Benchmark

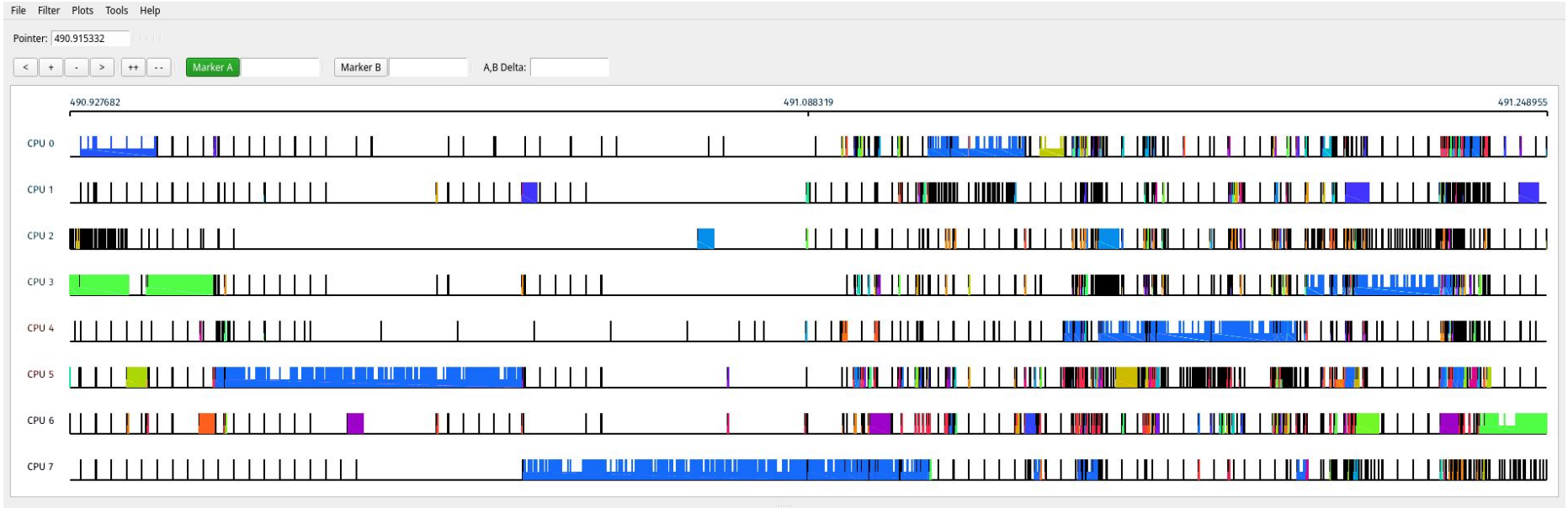


C-Ray Benchmark



EPC Implementation

Shared Memory Implementation



Current and Future Work

- Investigate the performance of shared memory
  - I tried using madvise(MADV_WILLNEED).
    - It did not improve performance.
    - However page fault and CPU usage pattern changed.
  - Test performance on Windows
    - Windows has different overcommit policies.
- atexit() problem
  - Processes can register functions to be called at process termination with Unix atexit() API.
  - If code generated by llvm-jitlink registers a function, it is called when terminating llvm-jitlink or llvm-jitlink-executor process.
  - But those registered functions are long gone. Their memory has been unmapped.
  - Crashes with SIGSEGV.

Current and Future Work

- ClangREPL integration
  - It uses a SelfExecutorProcessControl along with InProcessMemoryManager.
  - It is easy to replace that with SimpleRemoteEPC that controls a fork()-ed process.
  - However, LLJIT uses LLJIT::PlatformSupport instead for memory actions.
  - The default implementation GenericLLVMIRPlatformSupport assumes everything is in-process.

Thank you