KUBERNETES COMMUNITY DAYS SHENZHEN 2023

# 在安卓子系统CHRE平台中应用WebAssembly

李炯强

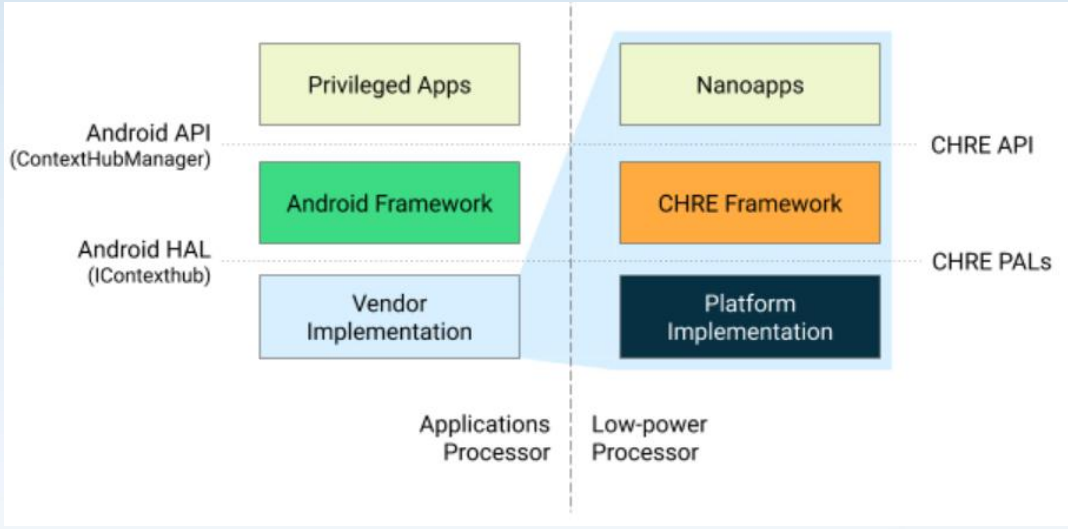
Content

# 目录

# Part 01
# 何为CHRE与Nanoapp

# 什么是CHRE与Nanoapp?

- CHRE是安卓开源项目（AOSP）的一部分，支持将计算从应用处理器迁移到小型低功耗处理器上。
- Nanoapp是一种运行在CHRE上的小型原生程序。是特定传感器部件的驱动或算法实现，且目前仅能使用C++编写。
- CHRE为支持Nanoapp管理多种传感器提供了丰富的基础API
- CHRE通过将Nanoapp内置其中或以动态链接库的形式进行加载以使得Nanoapp运行。
- 当前，Nanoapp在隔离性、跨架构以及跨平台方面存在局限性。这是由于Nanoapp被编译为原生指令集，并受制于原始设备制造商等因素所致。
- Wasm 可以帮助CHRE构建一个开放的生态系统!




得益于CHRE，用户将在快速配对功能上体验到更高的响应速度。

# Part 02
# 项目背景和意义

# 将Wasm应用于CHRE的意义

- 沙盒安全
- 跨平台与跨架构
- 使用多种语言开发Nanoapp的可能性
- 加载Nanoapp时，操作系统无关
- 构建开放生态系统

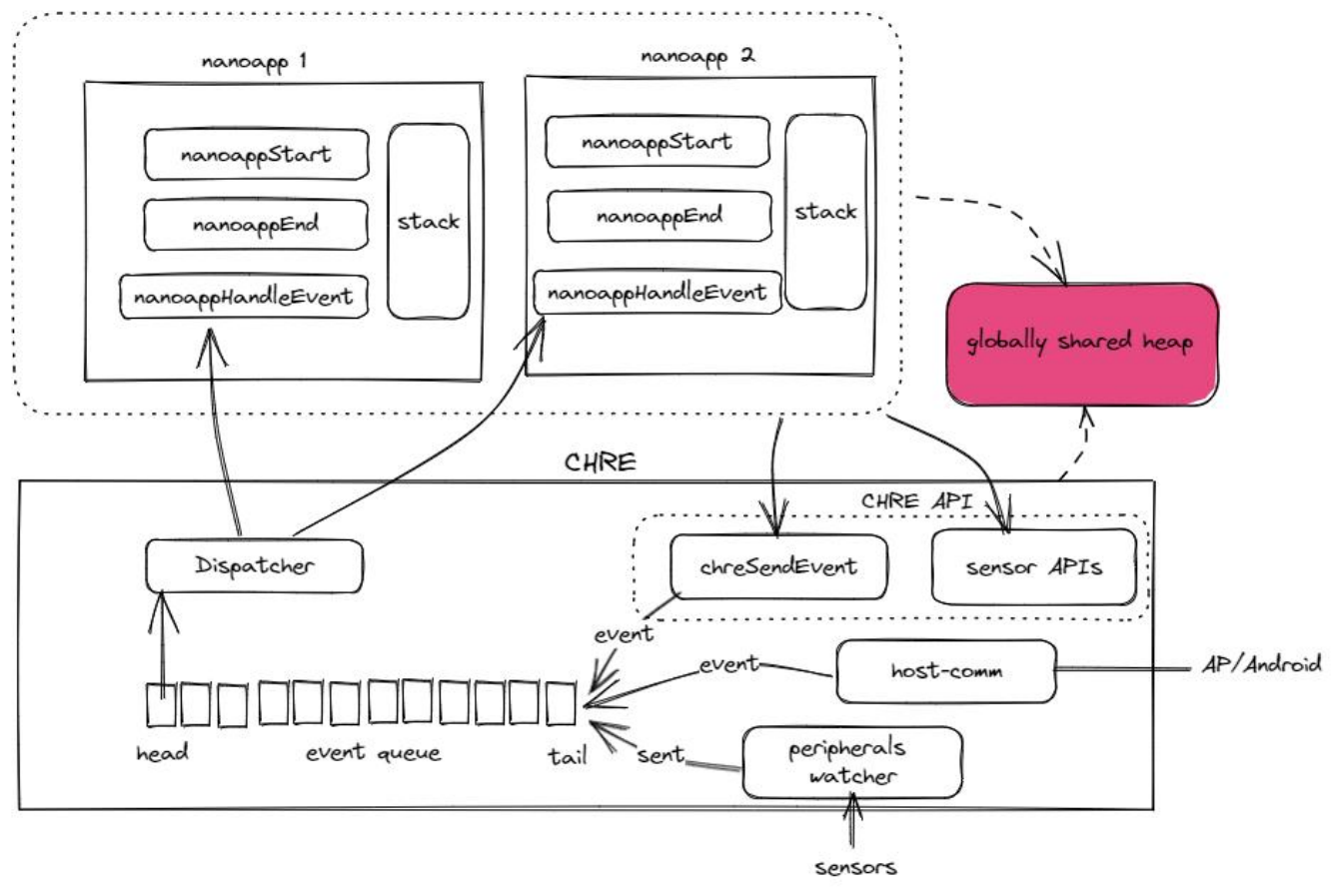
# CHRE-Wasm实验项目的背景

- 最终目的：将未修改的nanoapp源代码编译为Wasm，并在CHRE上运行。
- 设计目标：
  - 对CHRE内部代码和API不进行任何更改。
  - 保持现有的应用程序生命周期管理。
  - 原生Nanoapp和Wasm Nanoapp共存。
- 项目源码：
  - https://github.com/wasm-micro-runtime/chre-wasm
- 该项目使用了WebAssembly Micro Runtime
  - https://github.com/bytecodealliance/wasm-micro-runtime
  - C语言实现、低占用、高性能
  - 支持解释器模式、JIT(two compiler tiers, dynamic PGO)和AoT
  - 支持Wasi-libc(file, clock, socket, thread), Wasi-NN

# Part 03

# 将Wasm应用于CHRE中遇到的挑战

# CHRE 和 Nanoapp的工作流

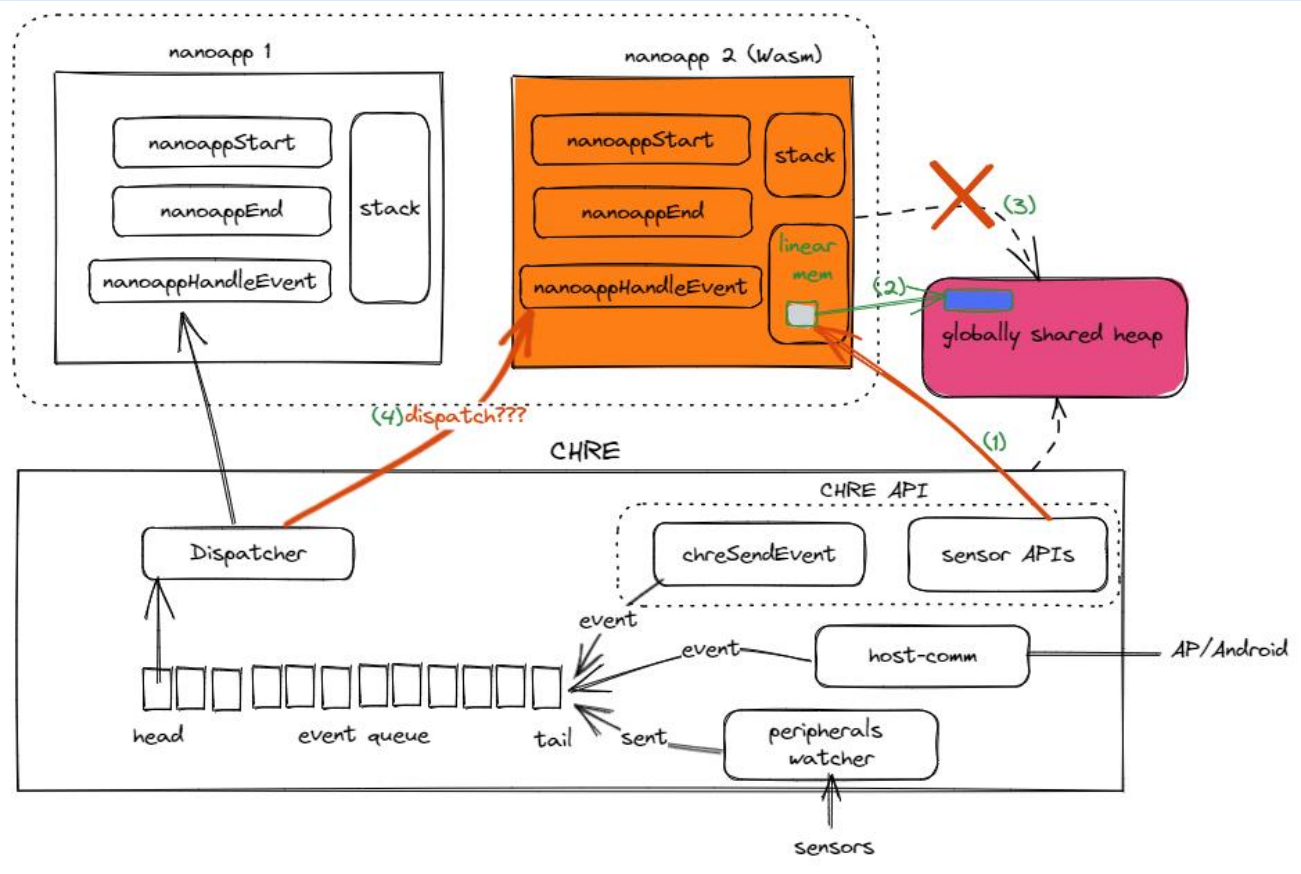



- CHRE维护一个全局事件队列，并调用目标Nanoapp的函数”nanoappHandleEvent”
- 事件可以来自Nanoapp、Host或传感器。
  - 事件数据是从全局堆分配的C结构体。
  - 事件发送者提供了freeCallback（用于释放事件的回调函数）。

```
bool chreSendEvent(uint16_t eventType, void *eventData,
                   chreEventCompleteFunction *freeCallback,
                   uint32_t targetInstanceId);
```

- Nanoapp使用指向应用程序堆栈或全局堆中的数据结构的参数来调用传感器API。
- 某些结构可能包含一个指针字段，用于保存CHRE返回的数据。

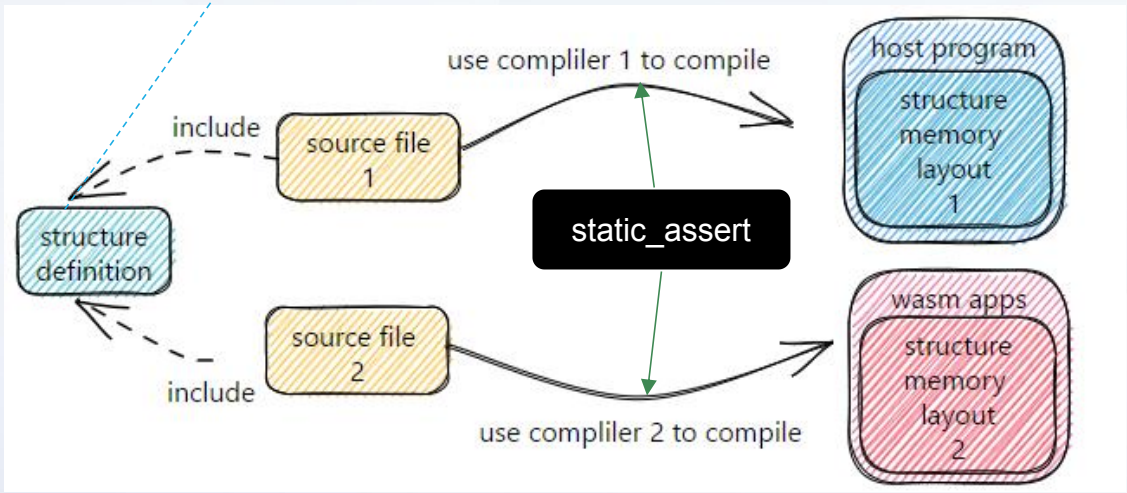
# 将Nanoapp迁移到Wasm的挑战




1. 确保Wasm和Native中的结构体定义具有相同的内存布局。
2. CHRE API参数中使用了多层指针。
3. Nanoapp（Wasm）无法访问传入事件中引用的数据。
4. 将事件数据分派给Wasm函数。

# 确保Wasm和Native中的结构定义具有相同的内存布局。

```
struct chreNanoappInfo {
    uint64_t appId;
    uint32_t version;
    uint32_t instanceId;
};
```




**通用解决方法:**

使用static_assert来确保Wasm world和native world中具有相同的内存布局。

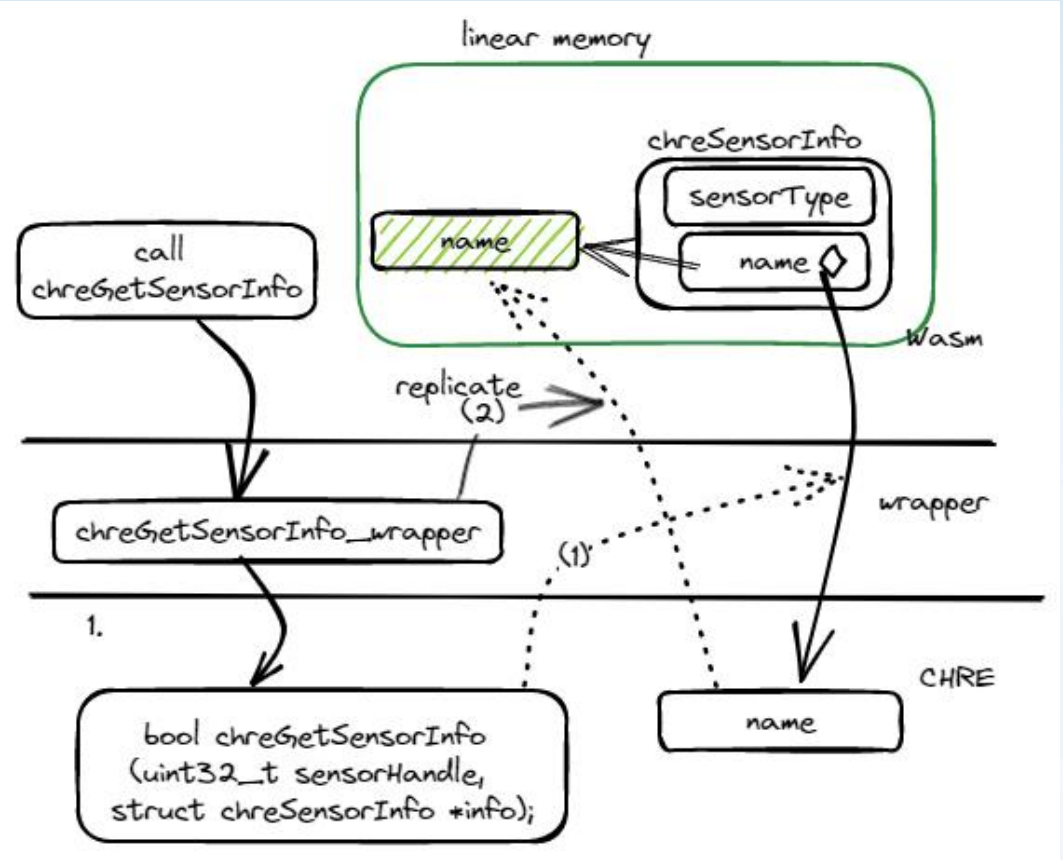
```
static_assert(sizeof(struct chreNanoappInfo) == 16
        && offsetof(struct chreNanoappInfo, appId) == 0
        && offsetof(struct chreNanoappInfo, version) == 8
        && offsetof(struct chreNanoappInfo, instanceId) == 12);
```

# CHRE API的参数中使用了多级指针

```
struct chreSensorInfo {
    /**
     * The name of the sensor.
     */
    const char *sensorName;

    /**
     * One of the CHRE_SENSOR_TYPE_* defines above.
     */
    uint8_t sensorType;
};

bool chreGetSensorInfo(uint32_t sensorHandle,
                       struct chreSensorInfo *info);
```




在native中引入了chreGetSensorInfo_wrapper来处理Wasm调用chreGetSensorInfo(..)。

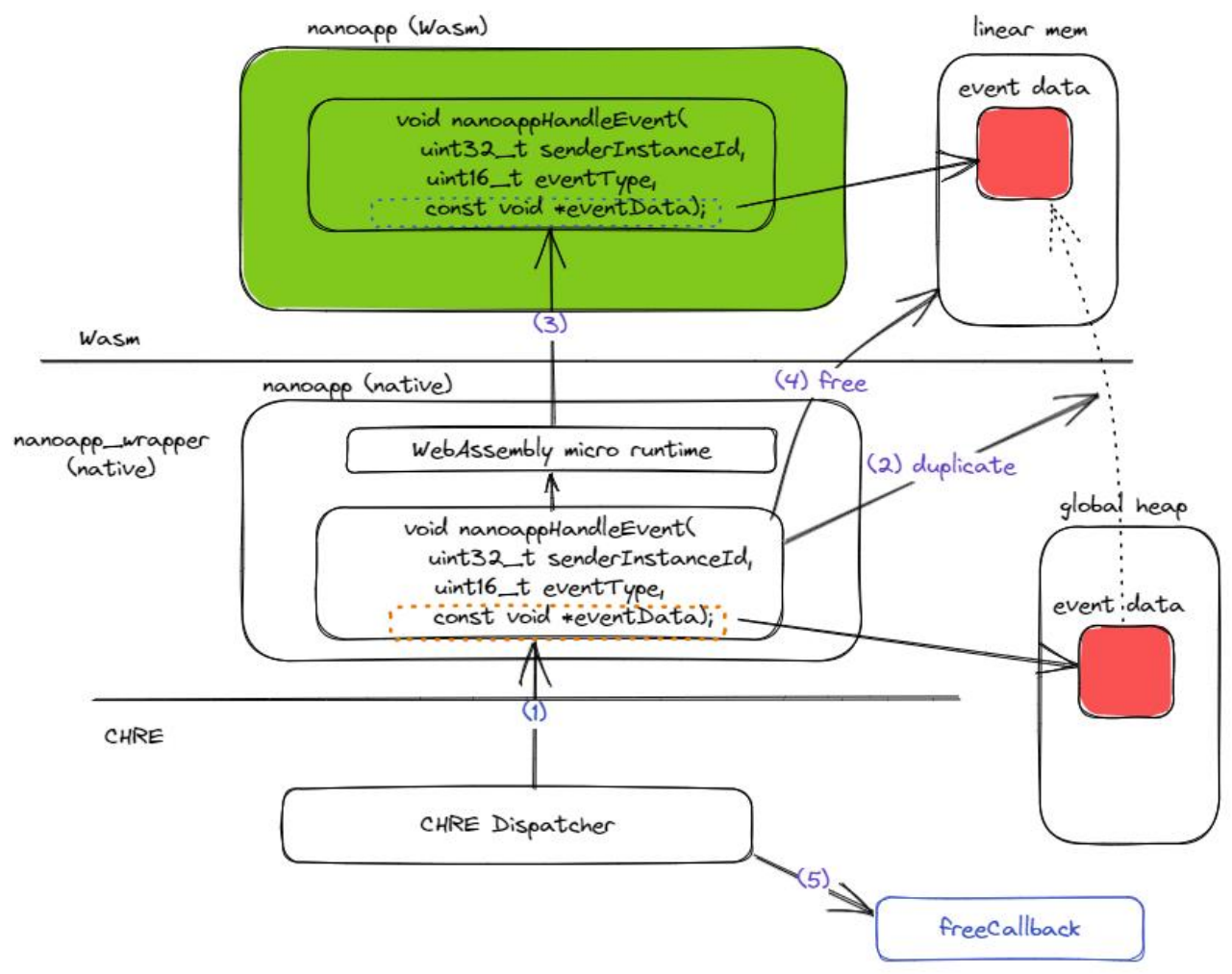
该适配函数将参数"info"的地址从Wasm线性内存偏移量转换为实际的地址，然后将执行权转交给原有的CHRE chreGetSensorInfo函数。

CHRE chreGetSensorInfo的实现将"sensorName"字段赋值为传感器名称字符串的地址。然而，Wasm是无法访问该地址的。

解决方法:

- chreGetSensorInfo_wrapper将在线性内存中复制"name"，并将字段重新赋值为线性内存中的"name"。
- 在线性内存中使用哈希表来防止重复复制”name”。

# 将事件从Native Nanoapp传递给Wasm Nanoapp进行处理




每一个Wasm Nanoapp都有一个与之关联的Native wrapper app

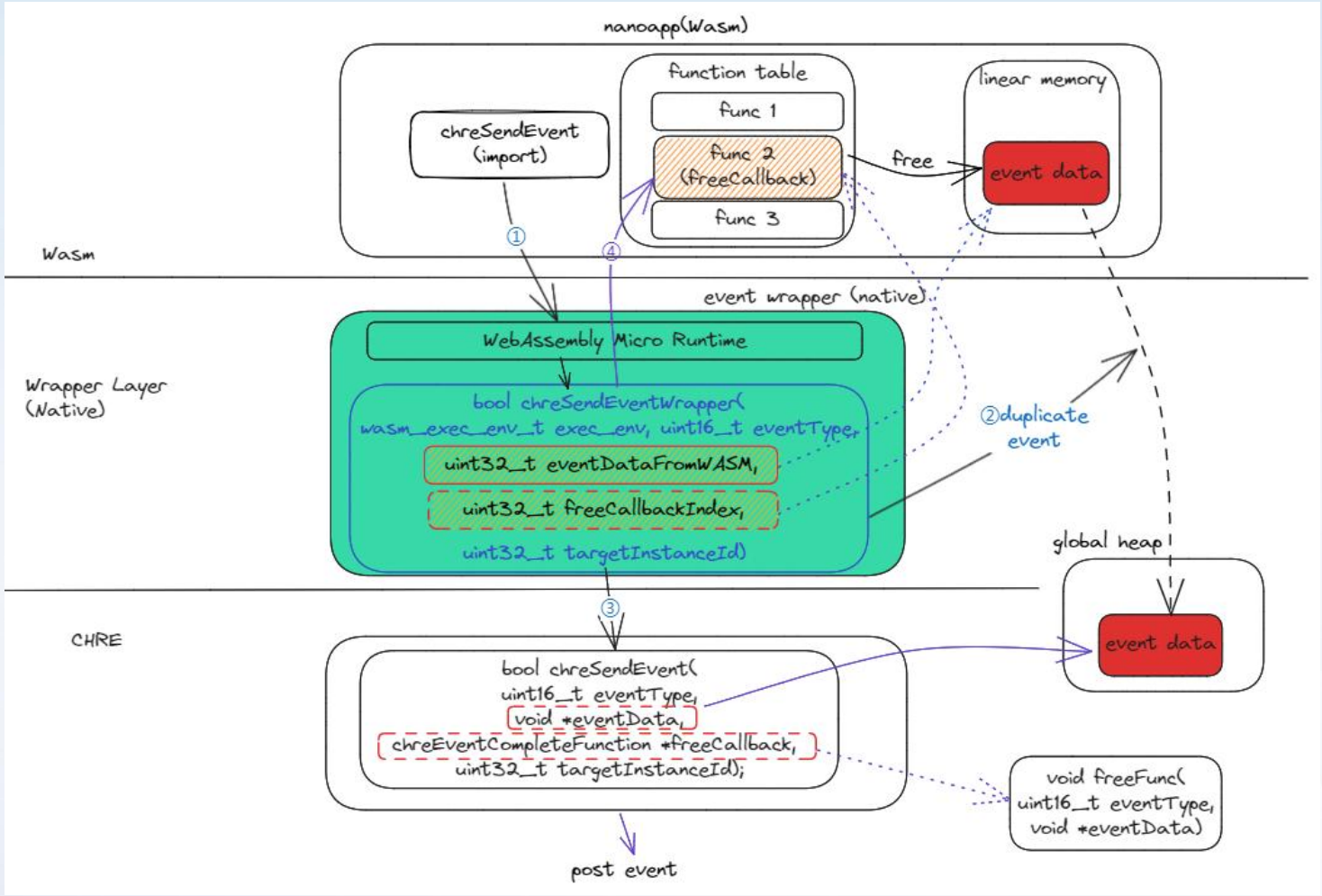
Native wrapper app中的nanoappHandleEvent函数由CHRE的事件分发器进行调用

Native nanoappHandleEvent函数必须从native的全局堆栈中复制事件数据到目标线性内存中

随后执行 Wasm nanoappHandleEvent并将复制到线性内存的事件数据传递

Native nanoappHandleEvent会负责释放拷贝的事件数据

# WasmNanoapp 发送事件




(1) Wasm Nanoapp使用参数eventData调用chreSendEvent，其中eventData指向线性内存，并使用Wasm freeCallback函数释放eventData。Native chreSendEventWrapper由WebAssembly Micro Runtime执行。参数freeCallback的值实际上是Wasm函数的索引。

(2) chreSendEventWrapper函数从全局堆栈中复制eventData，并保存为副本。

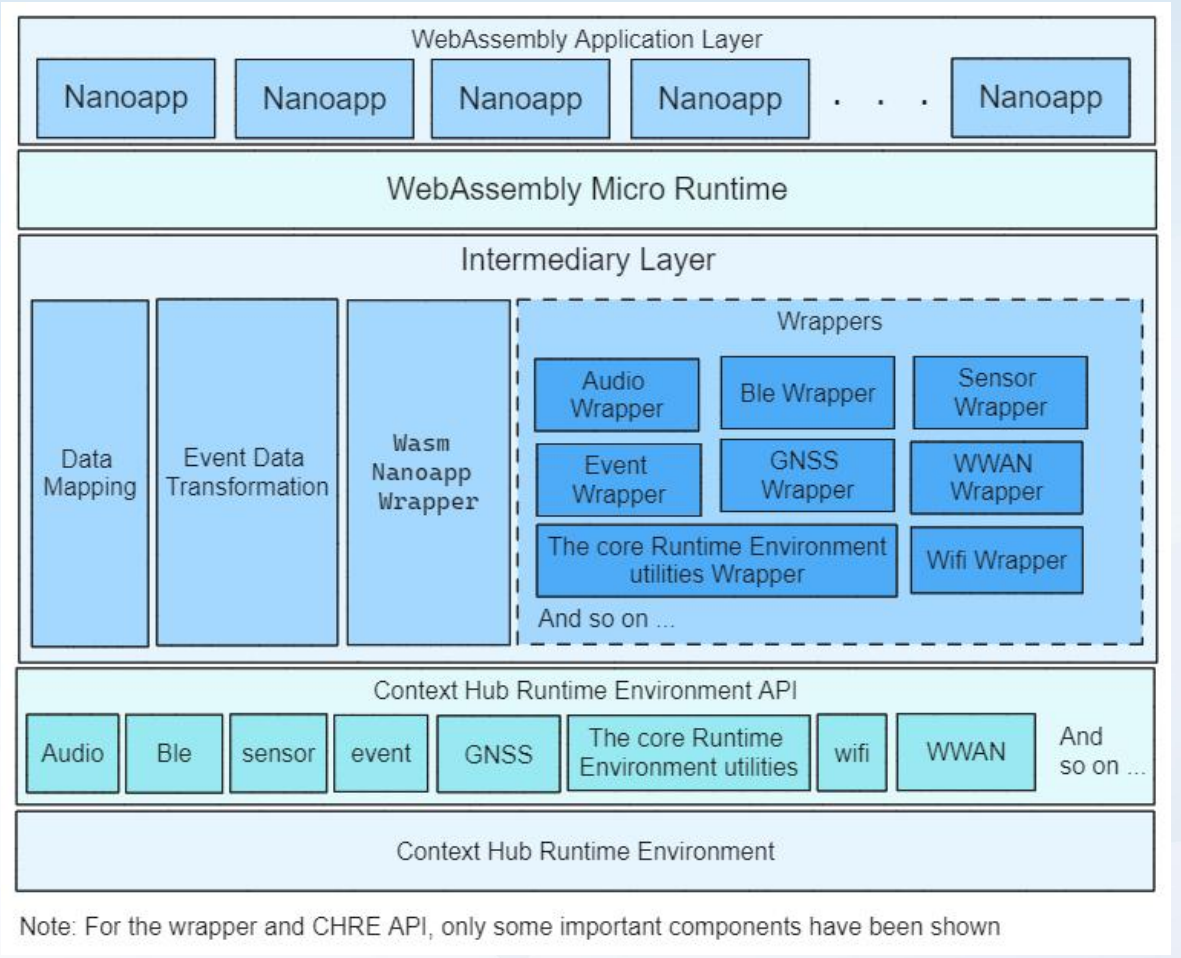
(3)然后调用真正的CHRE API chreSendEvent，其中eventData参数指向全局堆中复制的eventData，并添加一个本机的freeCallback（freeFunc）。

(4)调用由freeCallbackIndex表示的Wasm函数，以释放线性内存中的事件数据。

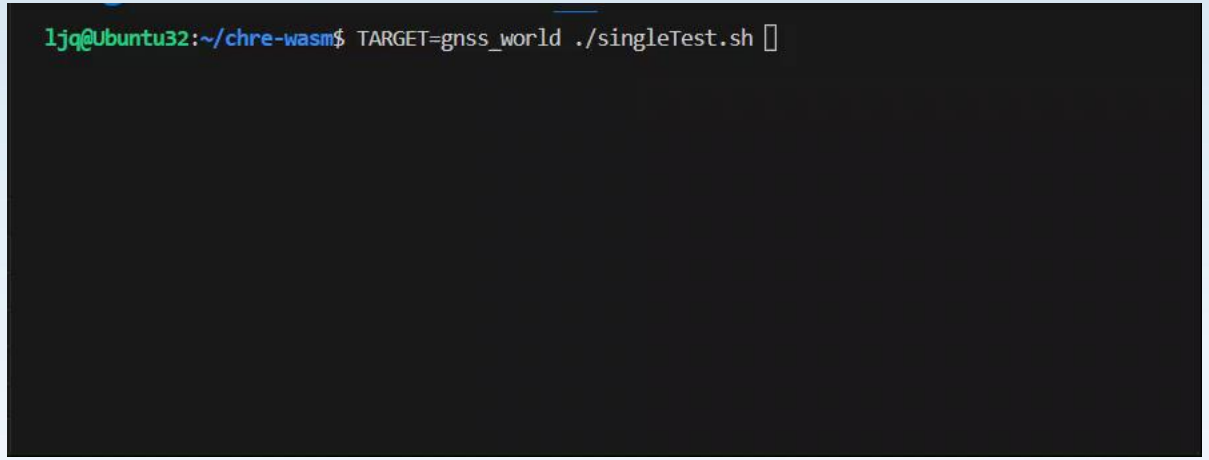
# Part 04
## CHRE-Wasm架构

# CHRE-Wasm 架构

- Wasm应用层: 跑nanoapp
- 中间层: (主要工作在这)
  - 数据映射层: 将沙盒外的内存映射到沙盒内的内存。
  - 事件数据转换：在沙盒内外之间转换各种事件。
  - Wasm Nanoapp适配层：支持CHRE管理和调用Wasm Nanoapp。
  - 适配层：支持Wasm Nanoapp调用CHRE API。



# Demo


在Linux上进行模拟，编译优化级别为3，删除符号信息。

二进制文件大小：在添加封装层和WAMR后增加了307kB。
内存使用：在已有的nanoapp DEMO上增加了2%至11%。

Demo：sensor_world 和 gnss_world

# Part 05
# 项目总结

# 项目总结

实现了项目目标和设计目标。

这些方法对于支持其他嵌入式框架上的Wasm应用程序（如LVGL）也可能有帮助。

然而，如果CHRE能够原生地支持Wasm Nanoapp，设计将会更加优雅。

在框架API设计中要考虑Wasm内存隔离的问题。

- 在native 框架中，使用句柄和API来访问Wasm内存
- 避免包含多层指针的结构体
- 使用消息序列化而不是共享数据结构的指针

CHRE-Wasm:
https://github.com/wasm-micro-runtime/chre-wasm

欢迎尝试使用 WebAssembly Micro Runtime:
https://github.com/bytecodealliance/wasm-micro-runtime

联系方式:
李炯强
bird@mail.ustc.edu.cn

# Thanks.