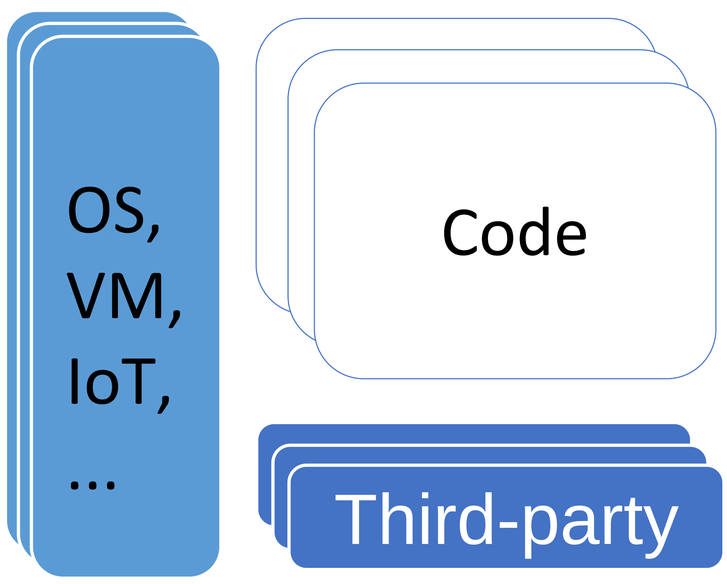
Memory-Usage Interfaces for Systems Code

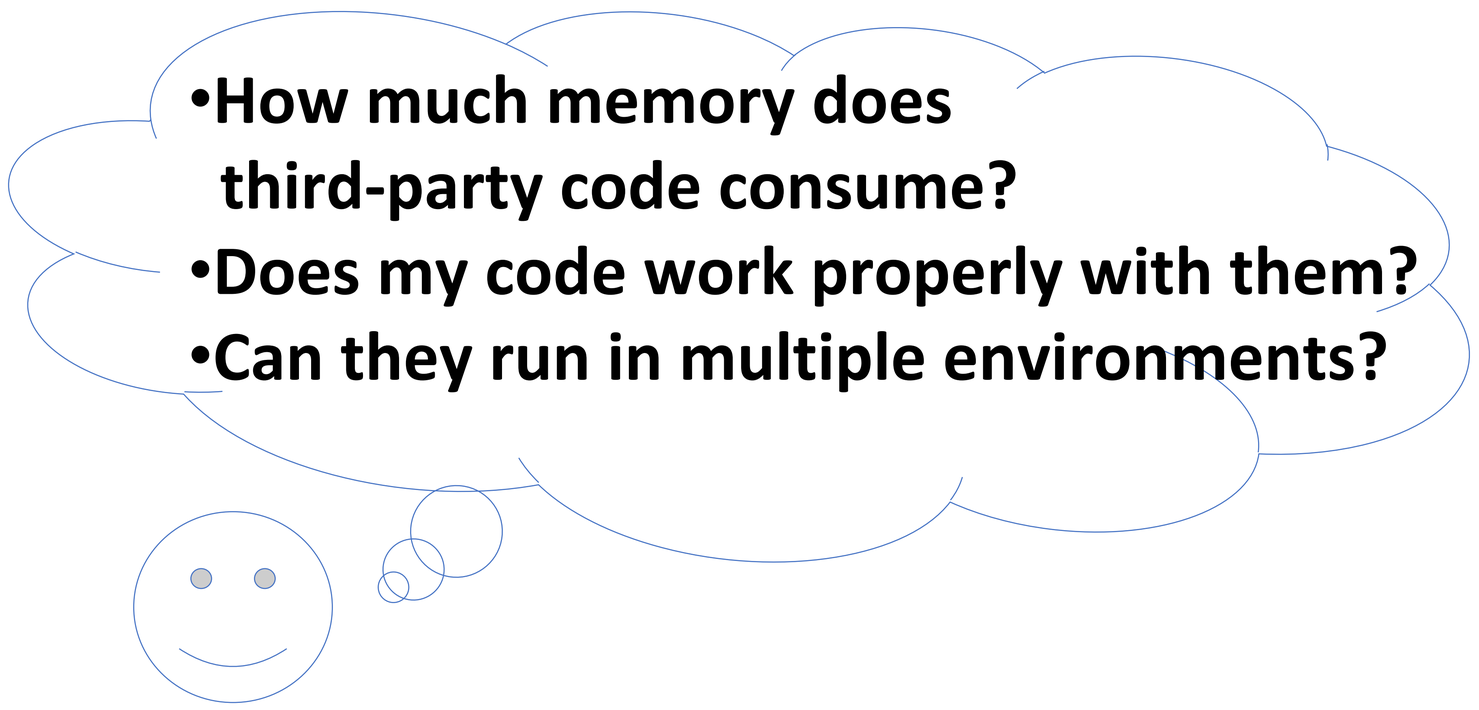
DSLab

Yonghao Zou

George Candea

There is no formal syntax for expressing memory usage
- Developers must implement and check memory usage carefully
- Third-party vendors cannot write clear memory usage instructions
- Operators need to deploy systems in different environments for testing





There are interfaces describing how to call the function properly, like the declaration with the function's parameters. Can we have memory-usage interfaces?

Functional interface

```
void *malloc(size);

void free(void *ptr);
```

```
int large_mem_func(n) {
 size = sizeof(..) * n;
 b = malloc(size, ..); ...
}
```

```
int many_alloc_func(int x) {
 malloc_object_a();
 malloc_object_b();
 release_a(); ... }
```

Memory-usage interface

```
[alloc size]
void *malloc(size);
[free resource_of(ptr)]
void free(void *ptr);
```

```
[alloc sizeof(..) * n]
int large_mem_func(n);
```

```
[alloc_and_free a b]
int many_alloc_func(int x);
```


With the memory usage interface
- Developers have a clear picture of memory usage for their code
- Third-party vendors can write clear memory usage instructions
- Operators can test and verify the systems before deployment

Want to work on something related? Talk to us!