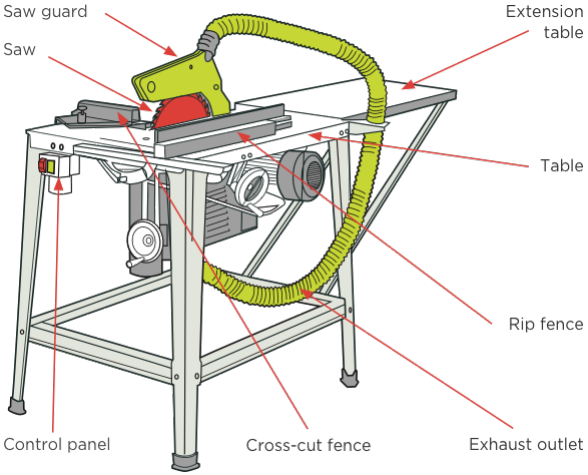
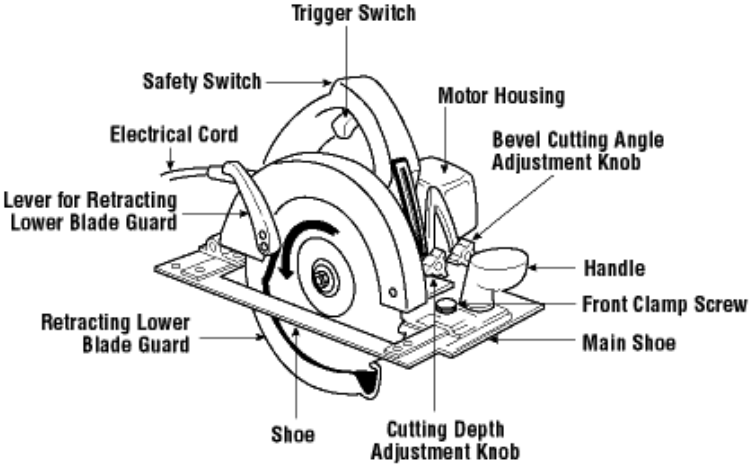
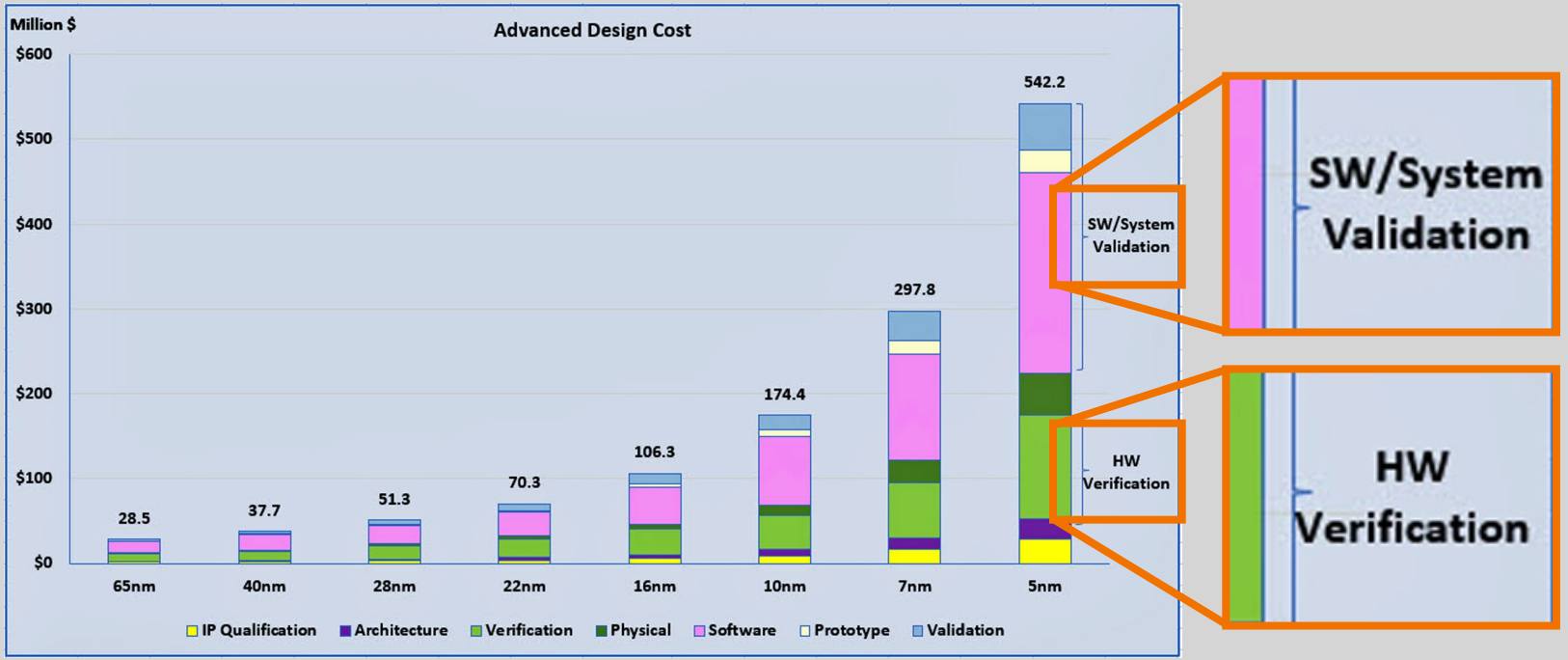
Exo: Let's Build Power Tools for Performance Engineers

Gilbert Bernstein, University of Washington

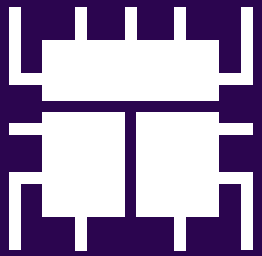

Saw guard
Saw
Extension table
Table
Rip fence
Control panel
Cross-cut fence
Exhaust outlet


Trigger Switch
Safety Switch
Motor Housing
Electrical Cord
Bevel Cutting Angle Adjustment Knob
Lever for Retracting Lower Blade Guard
Handle
Front Clamp Screw
Retracting Lower Blade Guard
Main Shoe
Shoe
Cutting Depth Adjustment Knob

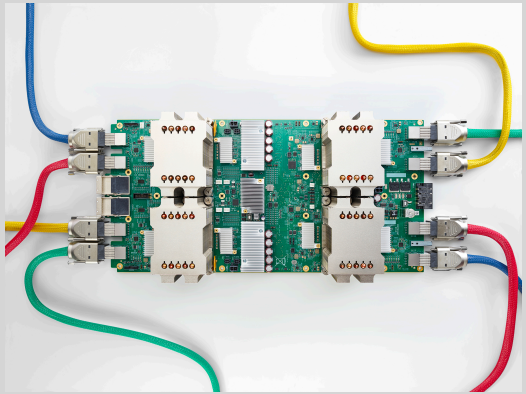
Verification & Validation Dominate HW Development Cost

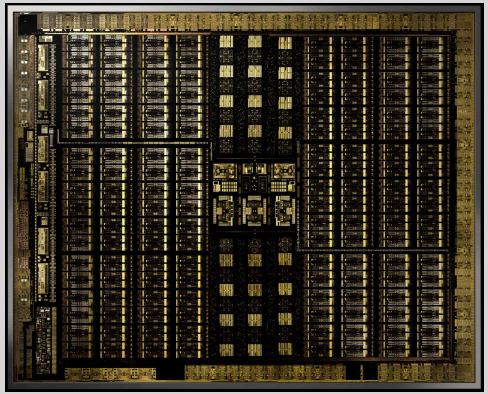


source: IBM, by way of
https://www.eeworldonline.com/hardware-assisted-verification-technology-in-the-mainstream/

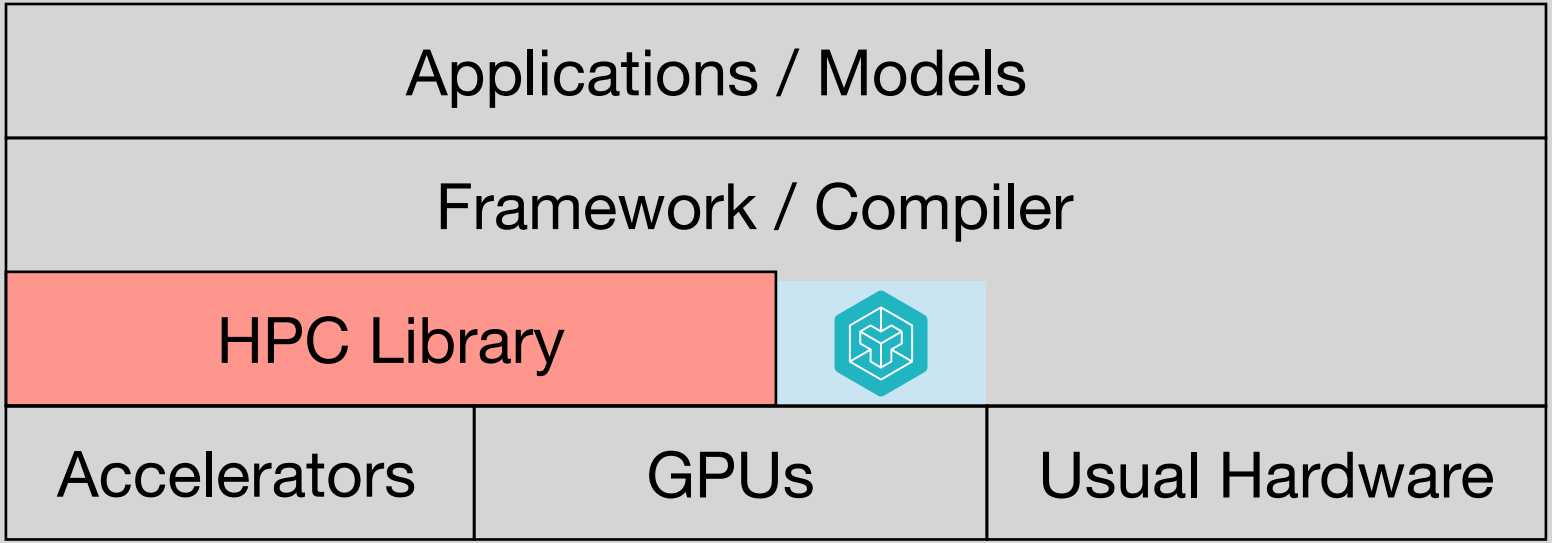


Exo

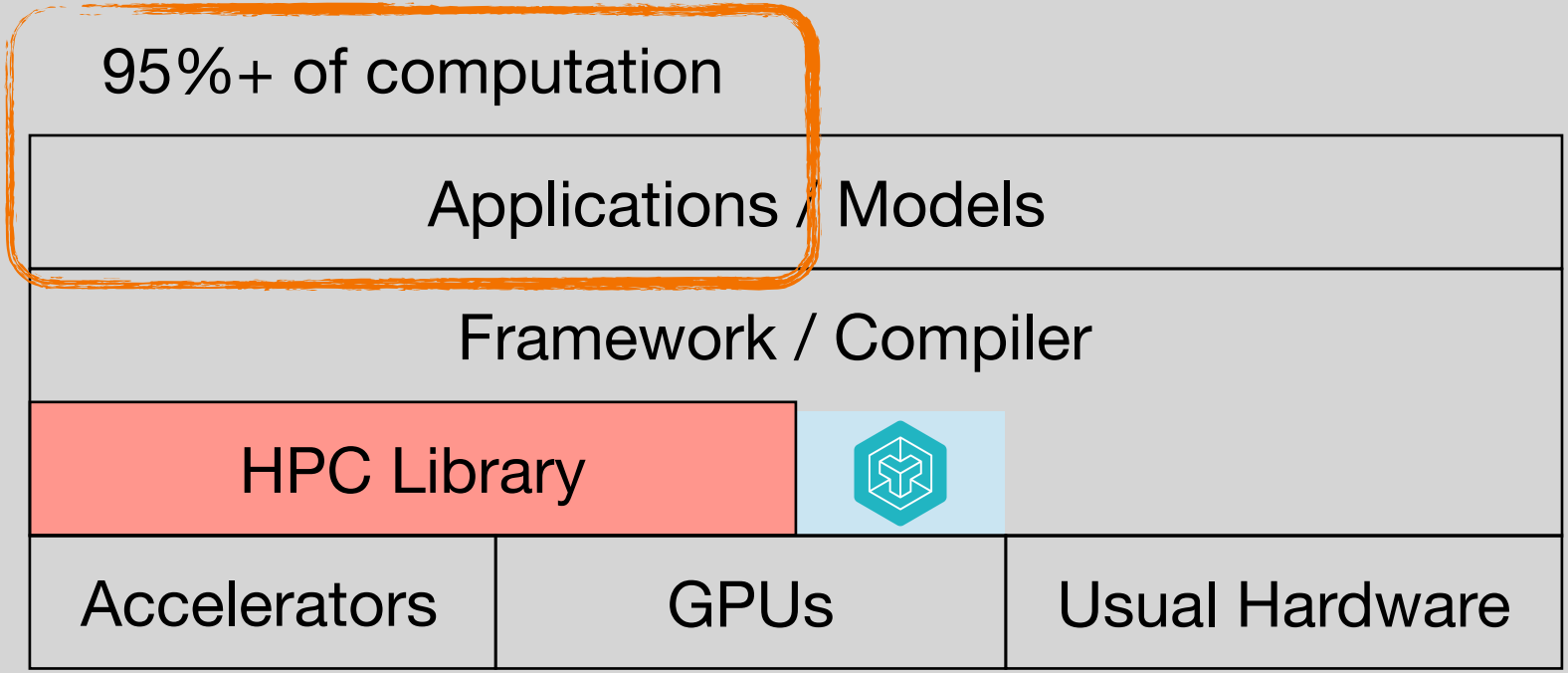




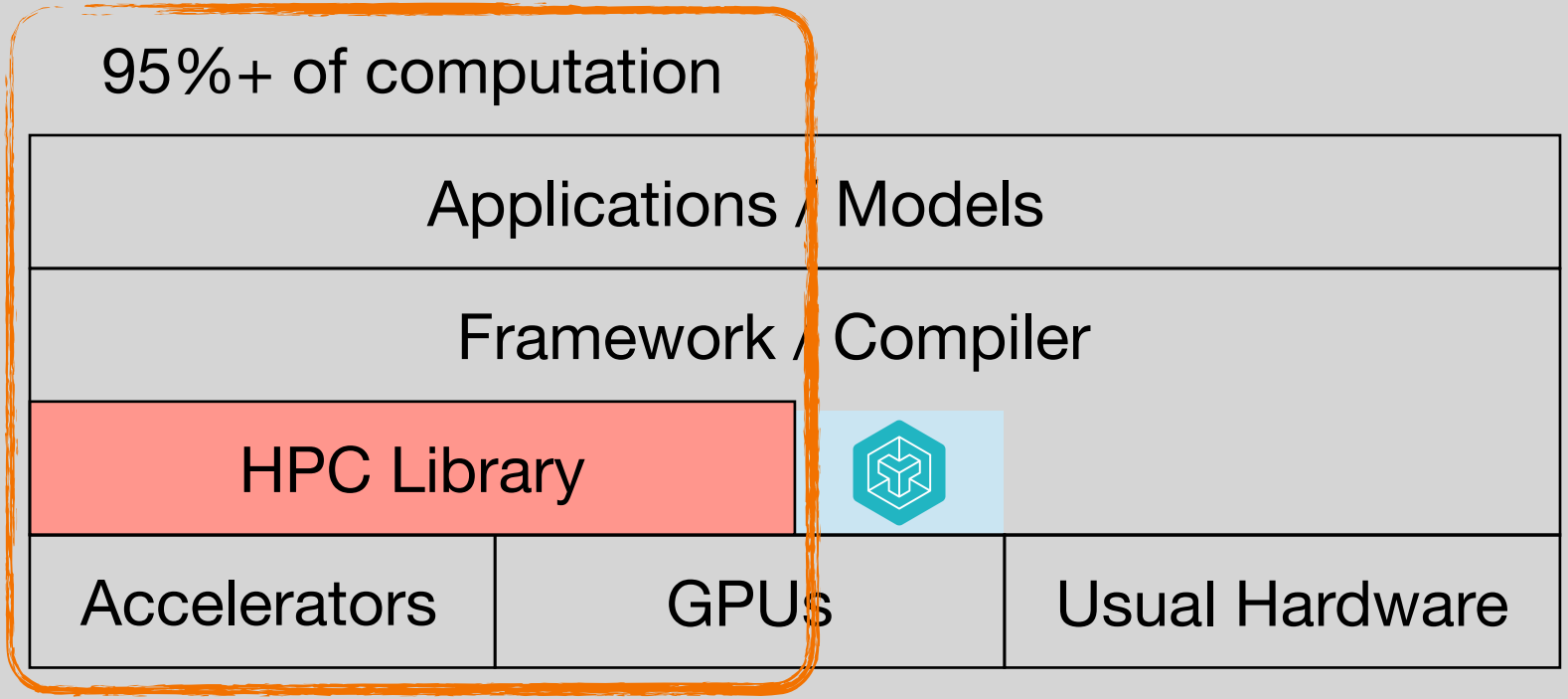
The ML Stack


Applications / Models
Framework / Compiler
HPC Library
Accelerators
GPUs
Usual Hardware

The ML ~~Stack~~ Stovepipe



The ML ~~Stack~~ Stovepipe



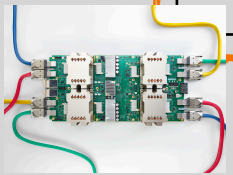
The ML ~~Stack~~ Stovepipe

95%+ of computation

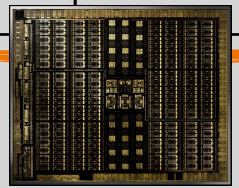
Applications / Models

Framework / Compiler

HPC Library



Accelerators

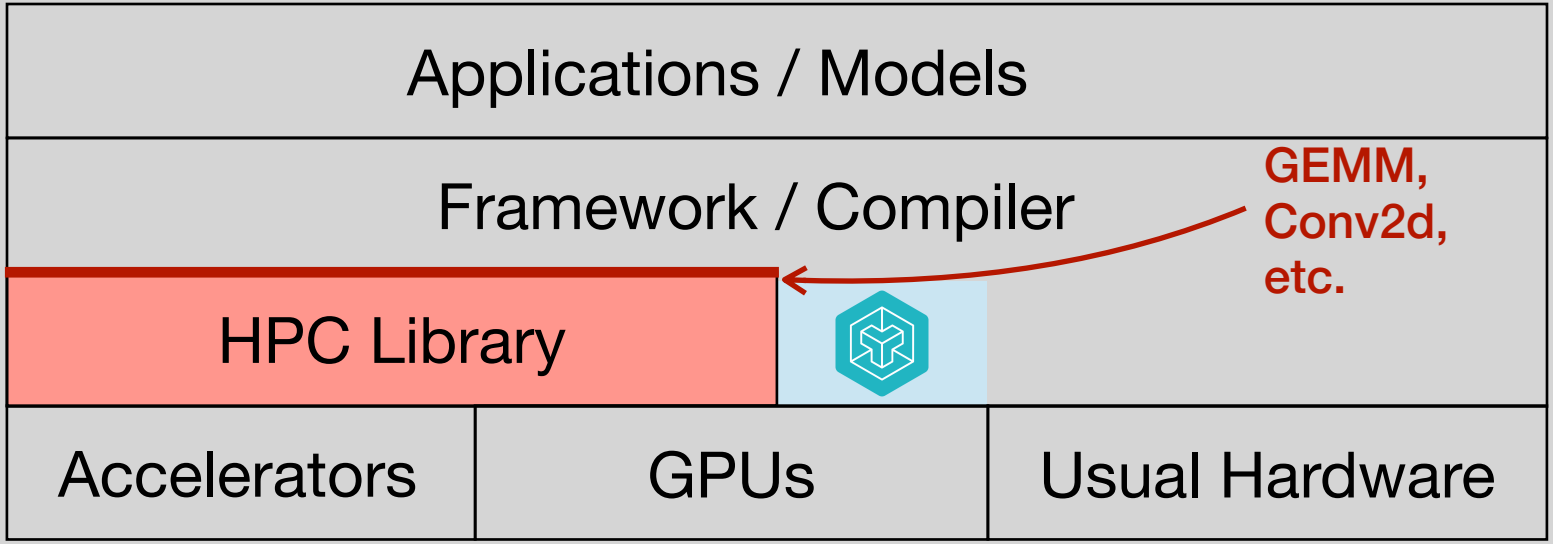


GPUs

Usual Hardware

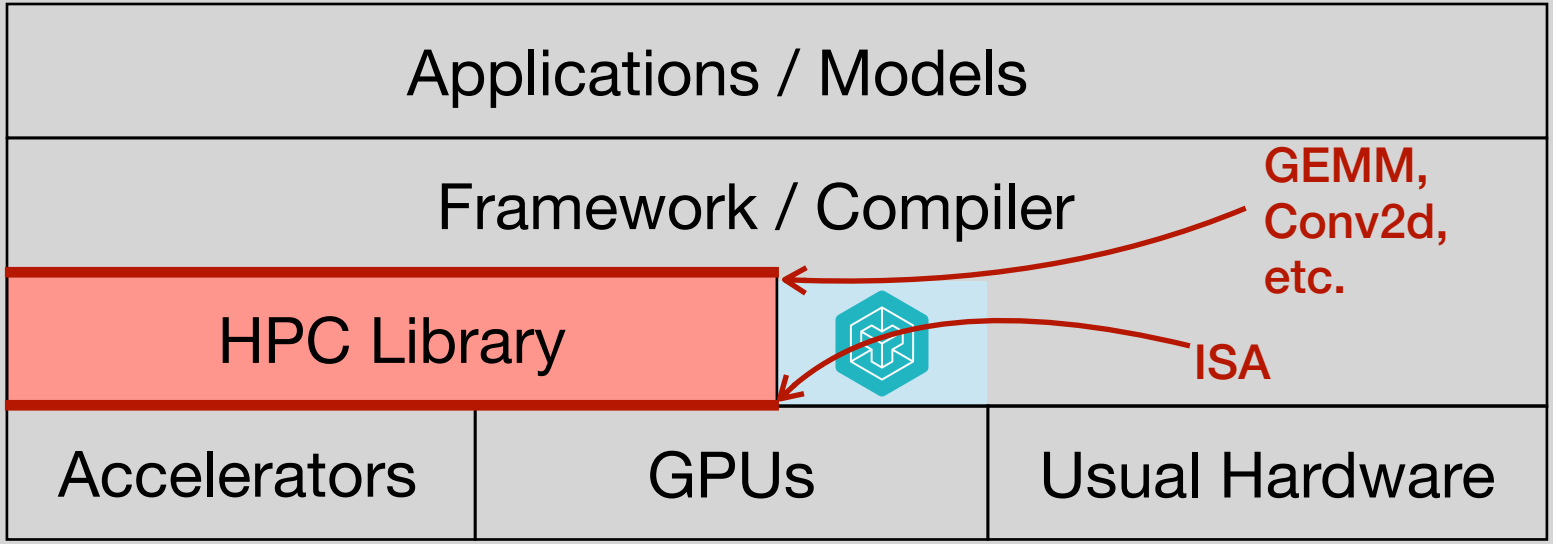
The ML ~~Stack~~ Stovepipe

Consequence 1: Portability Interface



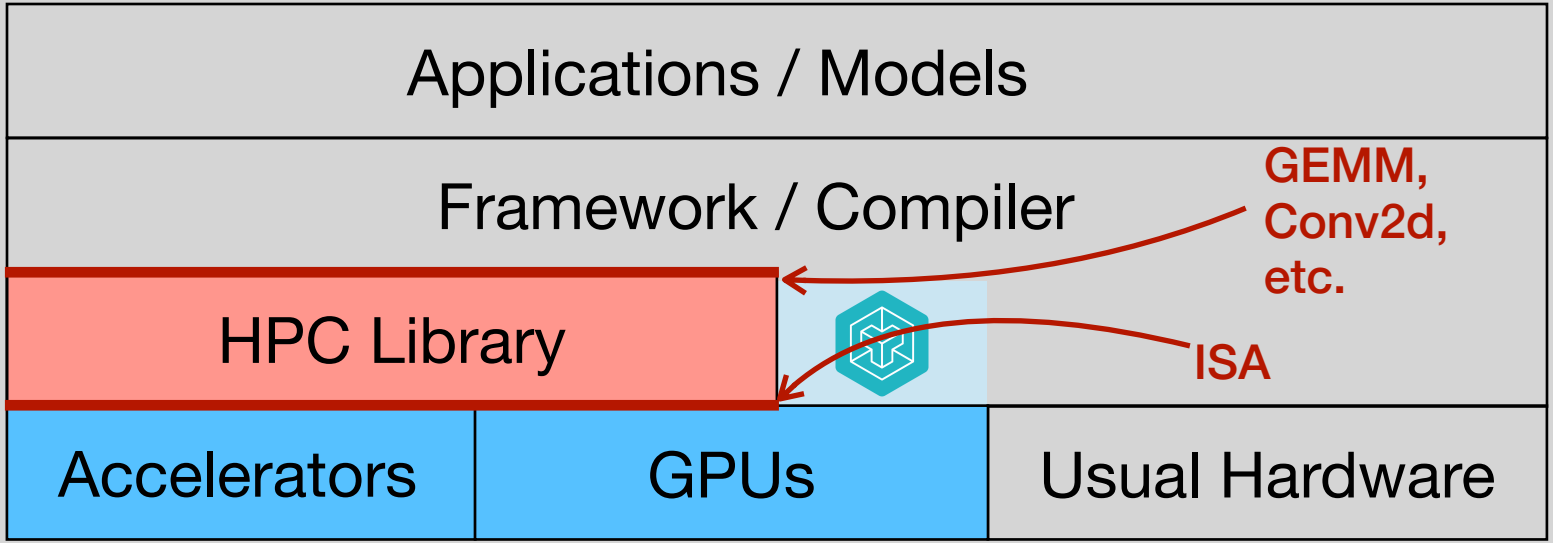
The ML ~~Stack~~ Stovepipe

Consequence 1: Portability Interface

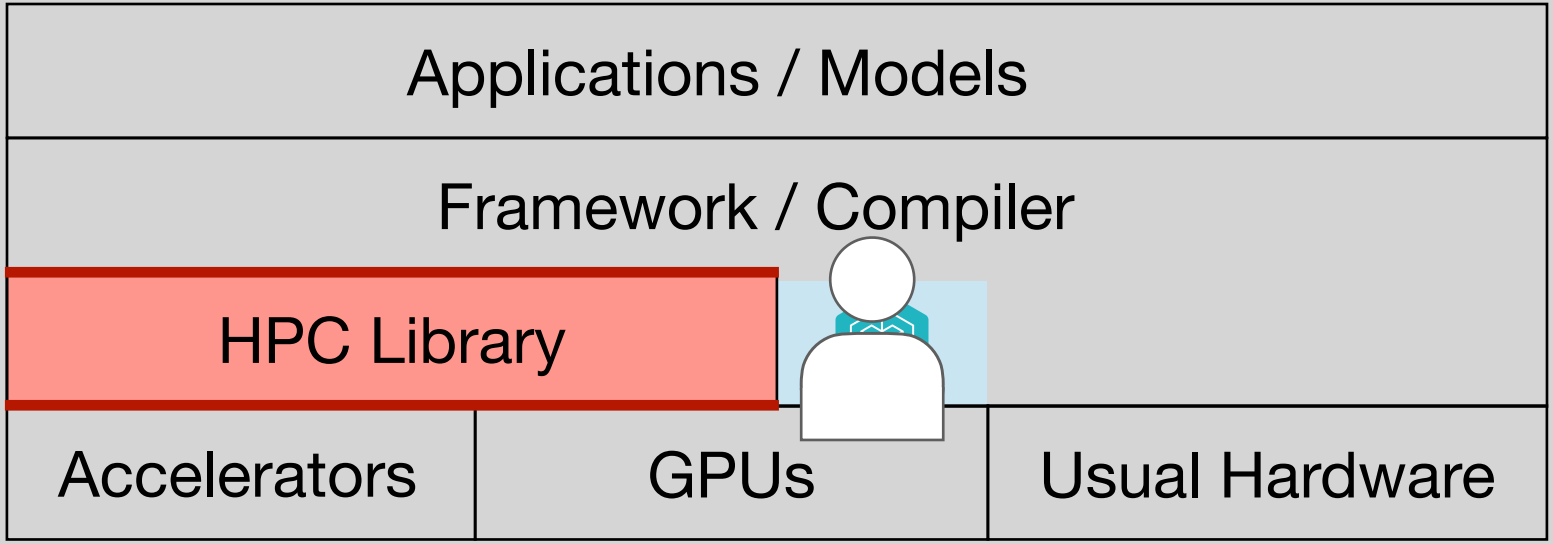


The ML ~~Stack~~ Stovepipe

Consequence 2: HPC Libs are on “Critical Path”



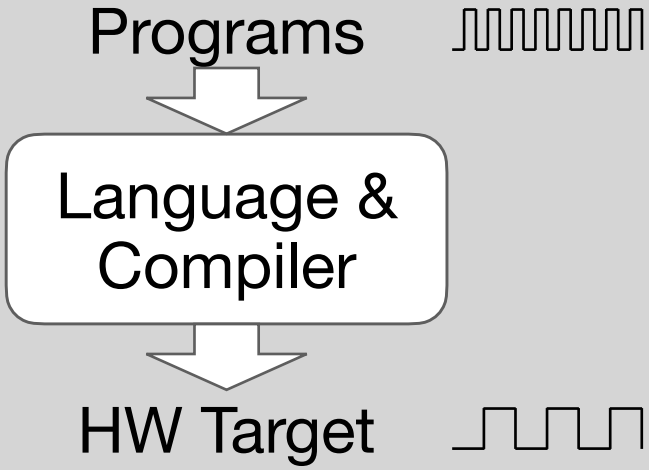
The ML ~~Stack~~ Stovepipe



What’s Special About Writing HPC Libraries?

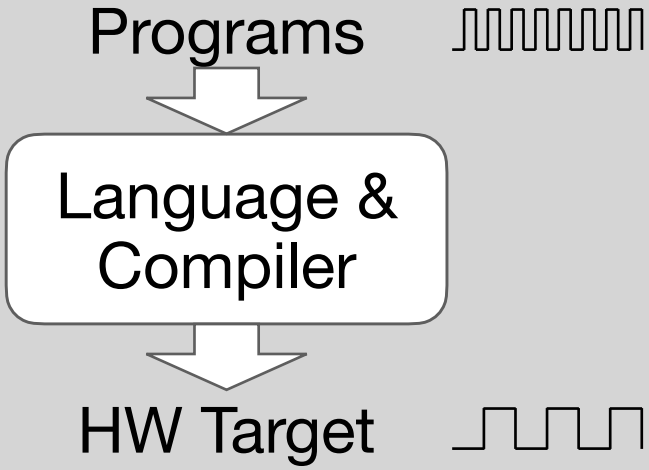
What's Special About Writing HPC Libraries?

Usual Programs

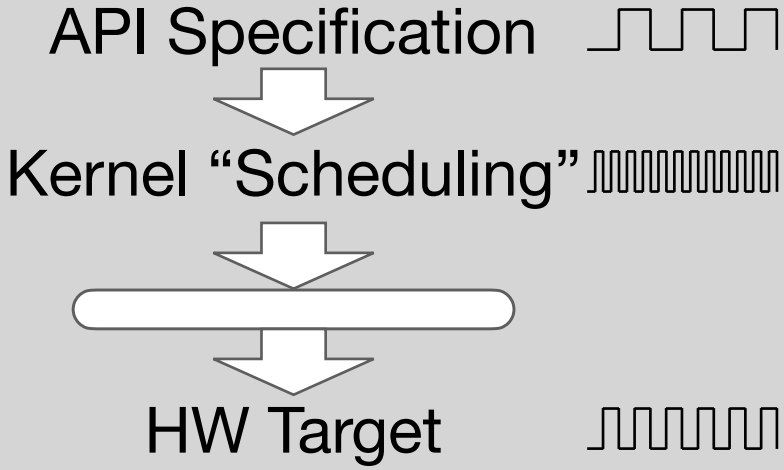


What's Special About Writing HPC Libraries?

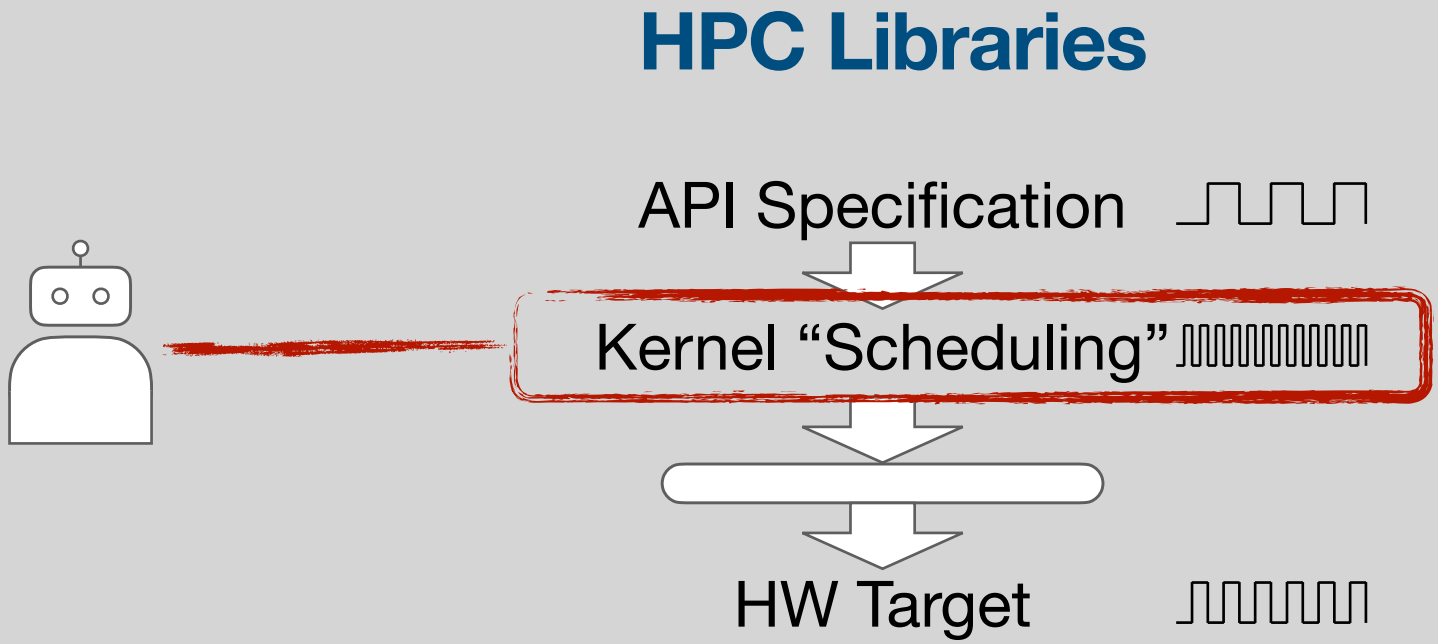
Usual Programs



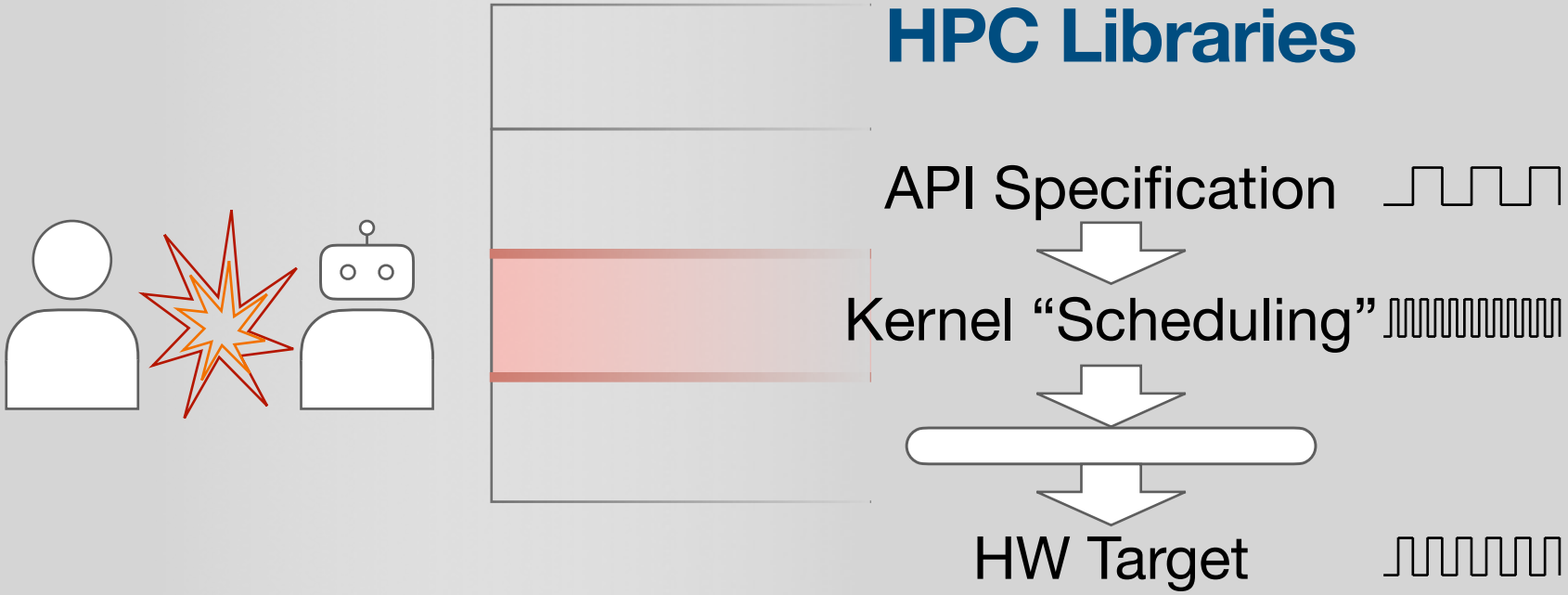
HPC Libraries



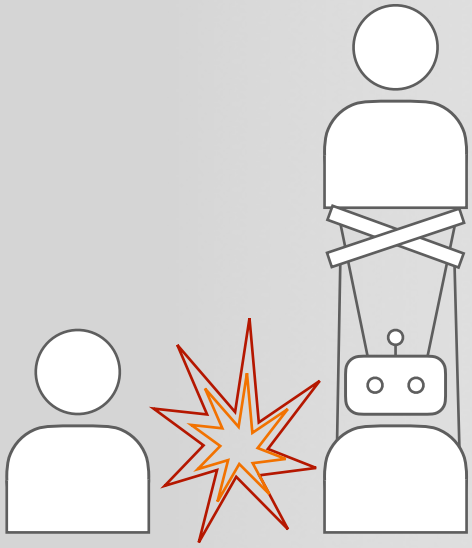
The Trouble With Automation



The Trouble With Automation



The Trouble With Automation



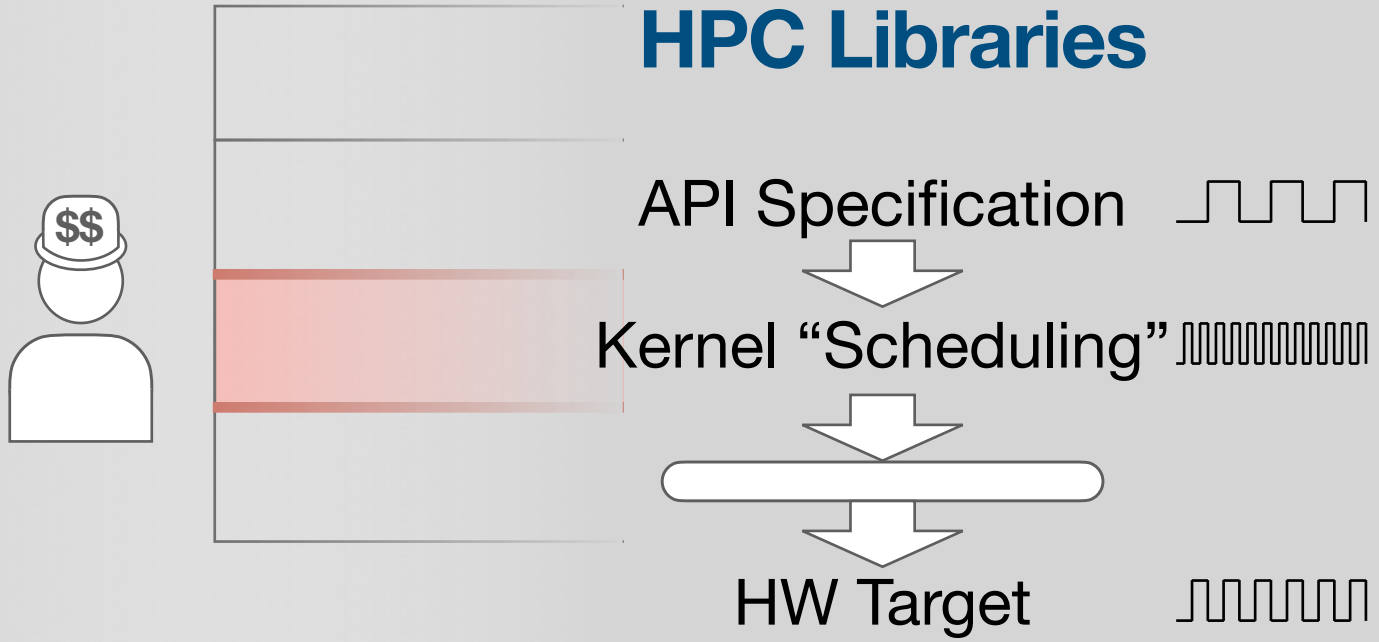
HPC Libraries

API Specification

Kernel “Scheduling”

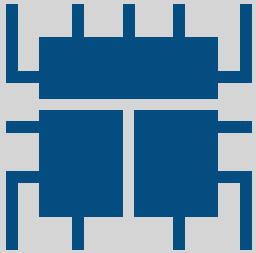
HW Target

The Trouble With Automation



Exocompilation

a compiler/language design that *externalizes* parts of the compiler in order to give programmers more control



Exo

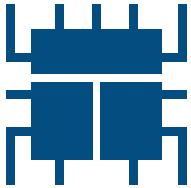
Exo code

```
def gemm(N : size,     M : size,     K : size,
         A : f32[N,K], B : f32[K,M], C : f32[N,M]):
    for i in seq(0,N):
        for j in seq(0,M):
            for k in seq(0,K):
                C[i,j] += A[i,k] * B[k,j]
```

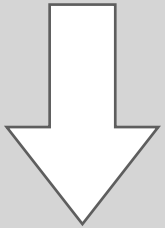
“basically C”

```
def gemm(N : size,     M : size,     K : size,
         A : f32[N,K], B : f32[K,M], C : f32[N,M]):
    for i in seq(0,N):
        for j in seq(0,M):
            for k in seq(0,K):
                C[i,j] += A[i,k] * B[k,j]
```

python™



Exo



```
void gemm(int N, int N, int K,
          float *A, float *B, float *C) {
    for (int i=0; i < N; i++) {
        for (int j=0; j < M; j++) {
            for (int k=0; k < K; k++) {
                C[i,j] += A[i,k] * B[k,j];
    }}}
```

Exocompilation

a compiler/language design that *externalizes* parts of the compiler in order to give programmers more control

Two Basic Ideas in Exo

Externalize Backends

User Scheduling

Exocompilation

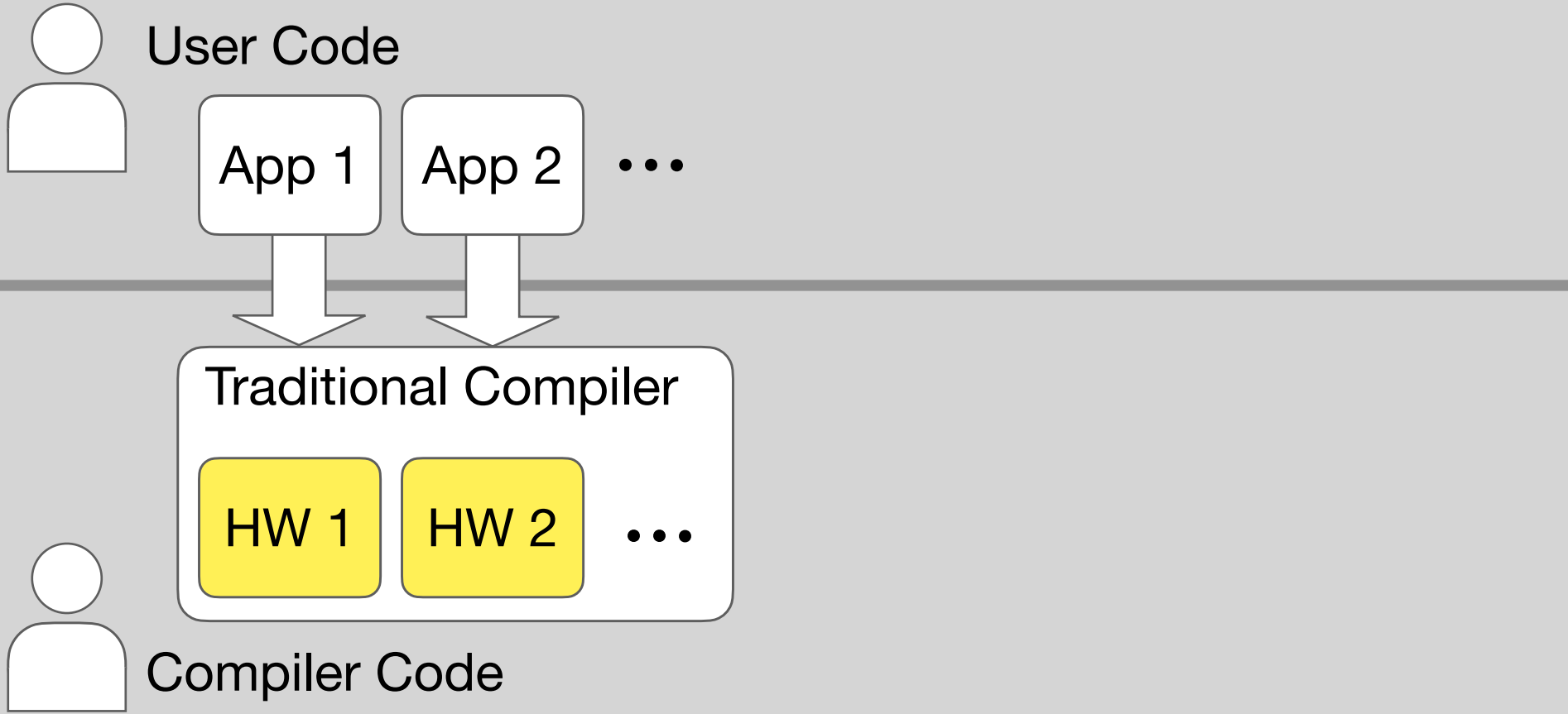
a compiler/language design that *externalizes* parts of the compiler in order to give programmers more control

Two Basic Ideas in Exo

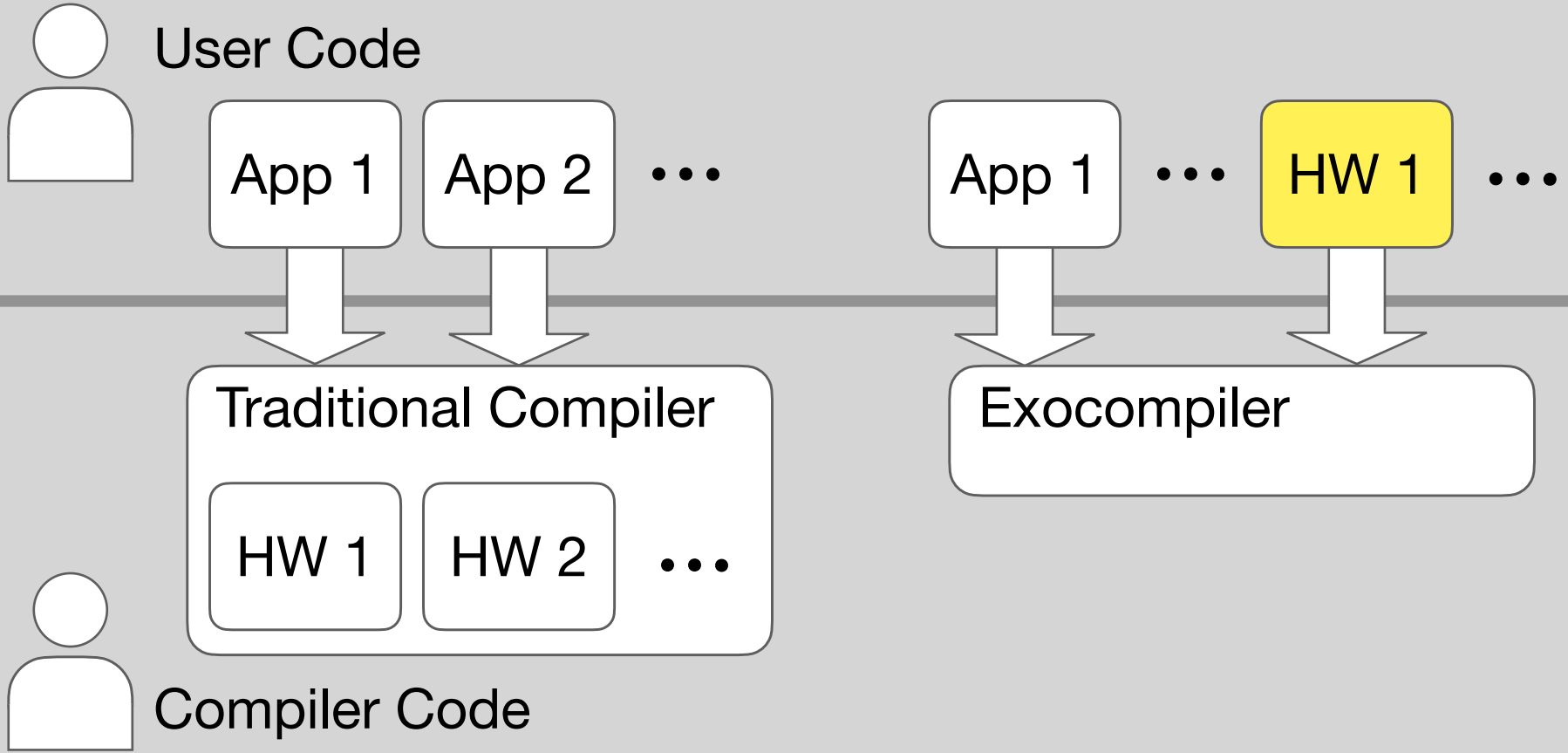
Externalize Backends

User Scheduling

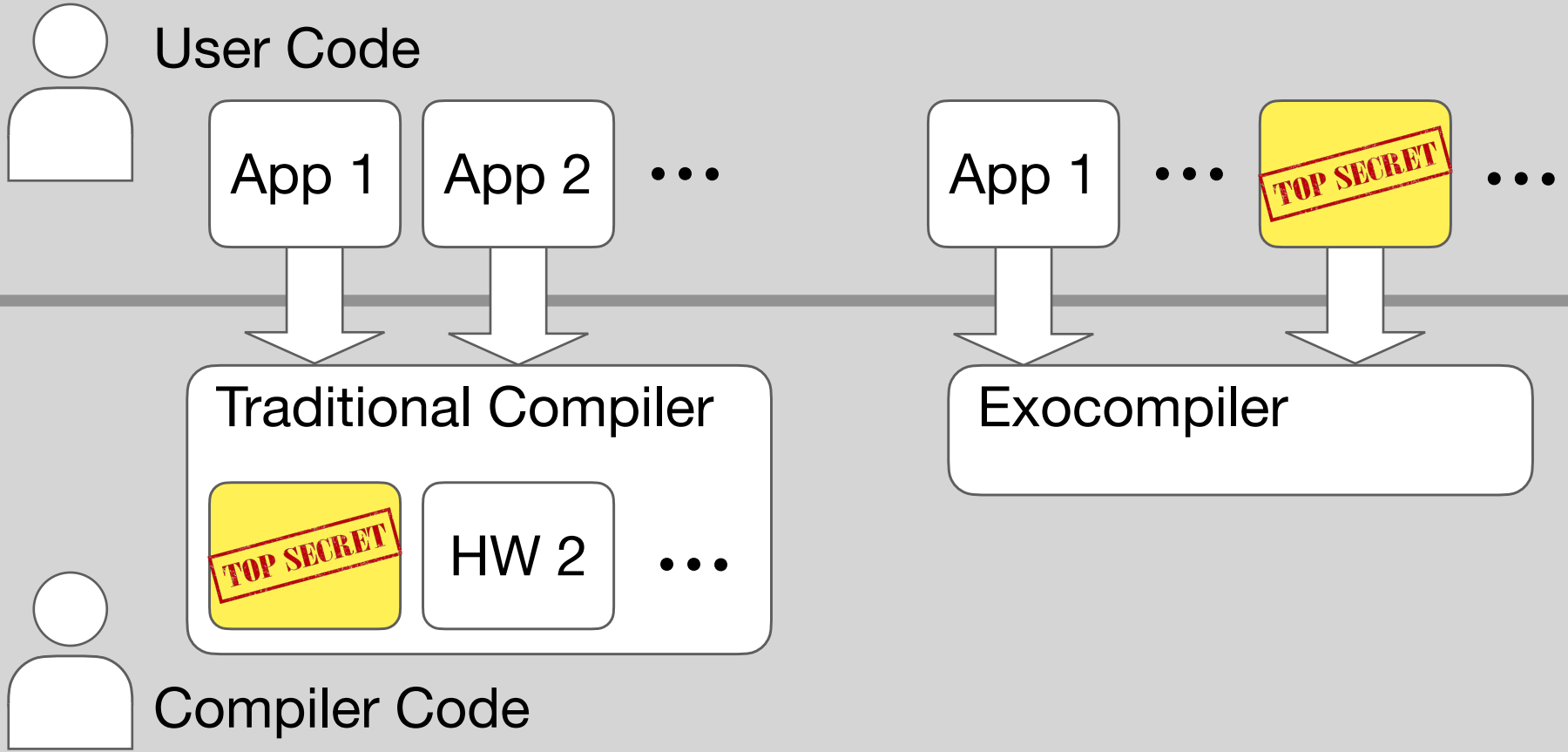
HW Backends as User-Level Libraries



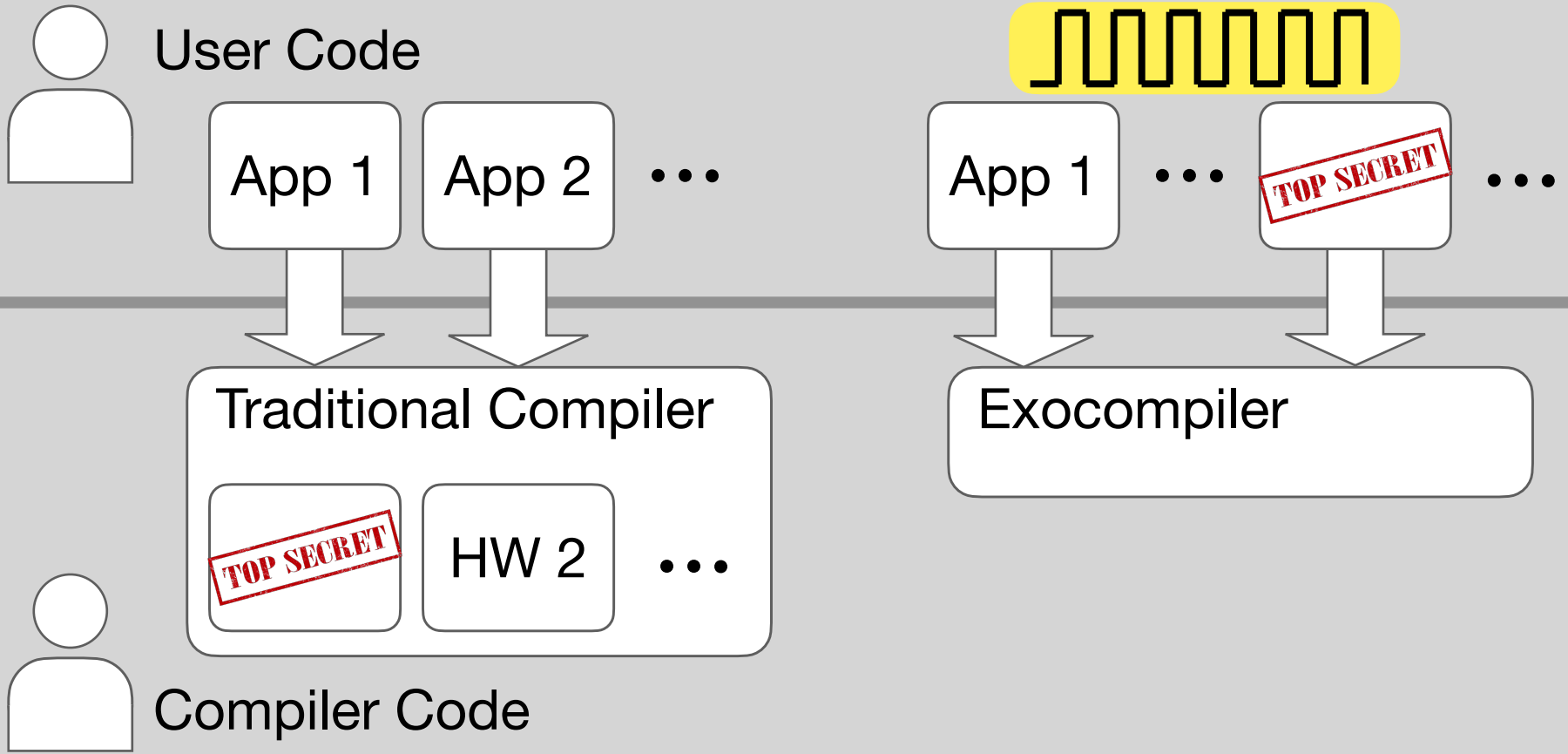
HW Backends as User-Level Libraries



HW Backends as User-Level Libraries



HW Backends as User-Level Libraries



Exocompilation

a compiler/language design that *externalizes* parts of the compiler in order to give programmers more control

Two Basic Ideas in Exo

Externalize Backends

User Scheduling

User-Schedulable Languages

```
def gemm(N : size,     M : size,     K : size,
         A : f32[N,K], B : f32[K,M], C : f32[N,M]):
    for i in seq(0,N):
        for j in seq(0,M):
            for k in seq(0,K):
                C[i,j] += A[i,k] * B[k,j]
```

User-Schedulable Languages

```
def gemm(N : size,     M : size,     K : size,
         A : f32[N,K], B : f32[K,M], C : f32[N,M]):
    for i in seq(0,N):
        for j in seq(0,M):
            for k in seq(0,K):
                C[i,j] += A[i,k] * B[k,j]
```

User-Schedulable Languages

```
def gemm(N : size,     M : size,     K : size,
         A : f32[N,K], B : f32[K,M], C : f32[N,M]):
    for i in seq(0,N):
        for j in seq(0,M):
            for k in seq(0,K):
                C[i,j] += A[i,k] * B[k,j]
```

User-Schedulable Languages

e.g.

CHiLL [’08]

Halide [SIGGRAPH ’12]

TACO [OOPSLA ’17]

LIFT [CGO ’17]

TVM [OSDI ’18]

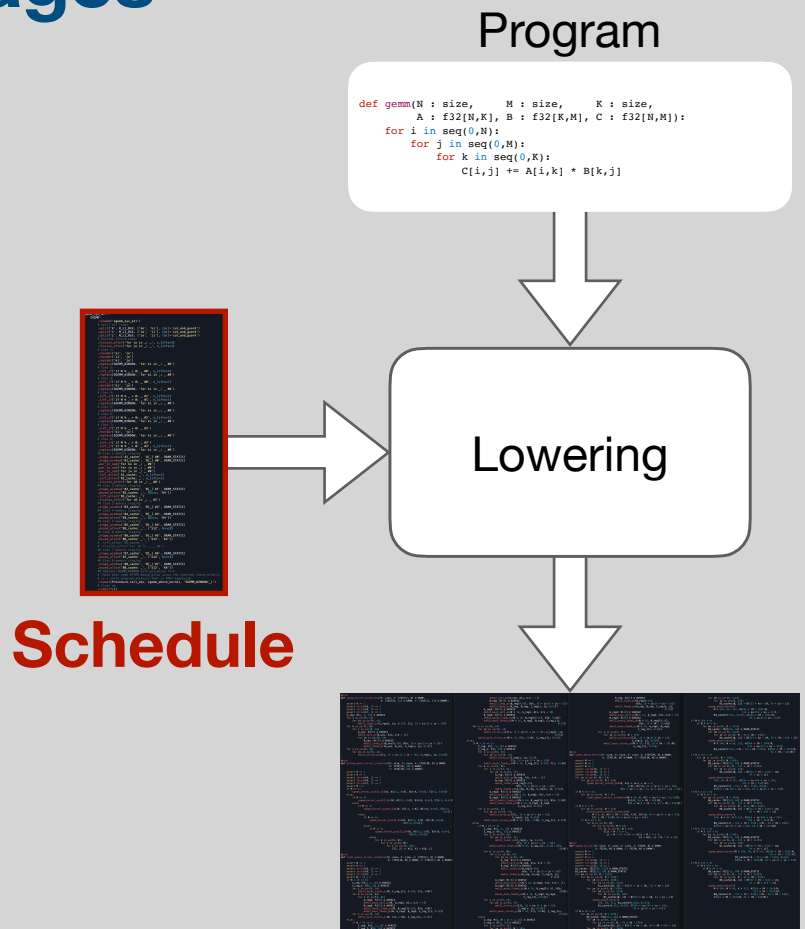
Tiramisu [CGO ’19]

```
def gemm(N : size,     M : size,     K : size,
         A : f32[N,K], B : f32[K,M], C : f32[N,M]):
    for i in seq(0,N):
        for j in seq(0,M):
            for k in seq(0,K):
                C[i,j] += A[i,k] * B[k,j]
```

User-Schedulable Languages

e.g.

- CHiLL [’08]
- Halide [SIGGRAPH ’12]
- TACO [OOPSLA ’17]
- LIFT [CGO ’17]
- TVM [OSDI ’18]
- Tiramisu [CGO ’19]



User-Schedulable Languages

e.g.

CHiLL [’08]

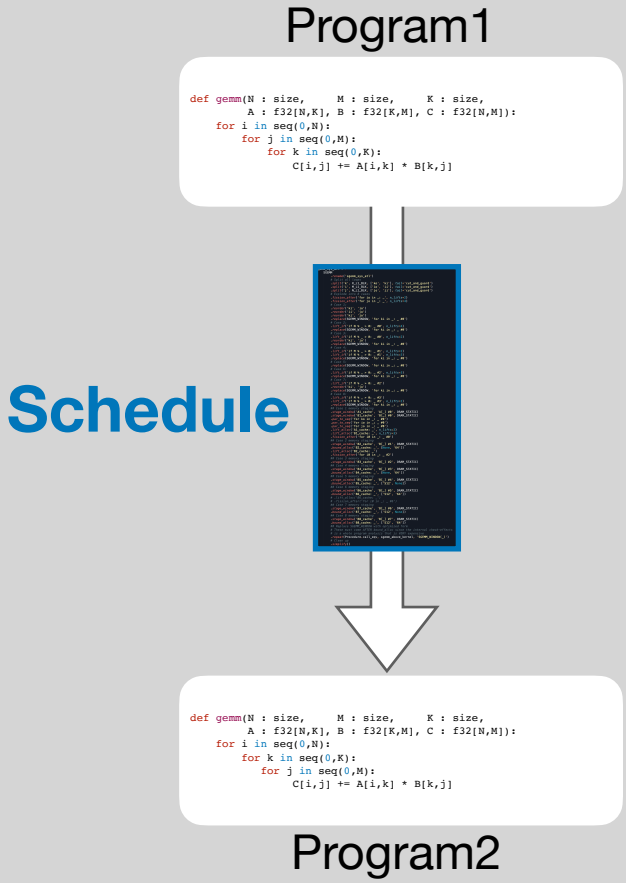
Halide [SIGGRAPH ’12]

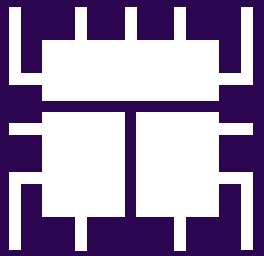
TACO [OOPSLA ’17]

LIFT [CGO ’17]

TVM [OSDI ’18]

Tiramisu [CGO ’19]





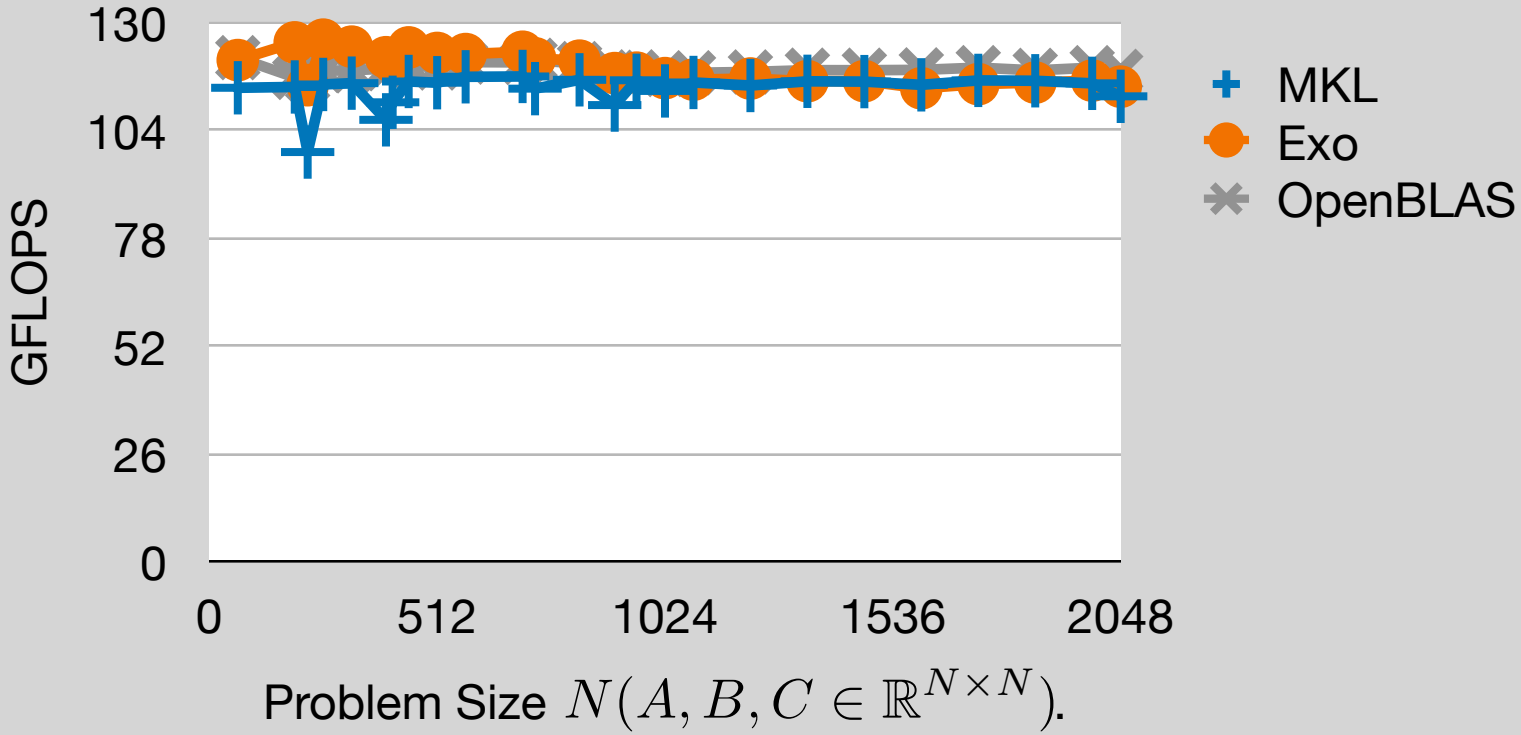
Exo Demo

Some Performance Numbers

AVX-512 MatMul

Matches best possible performance

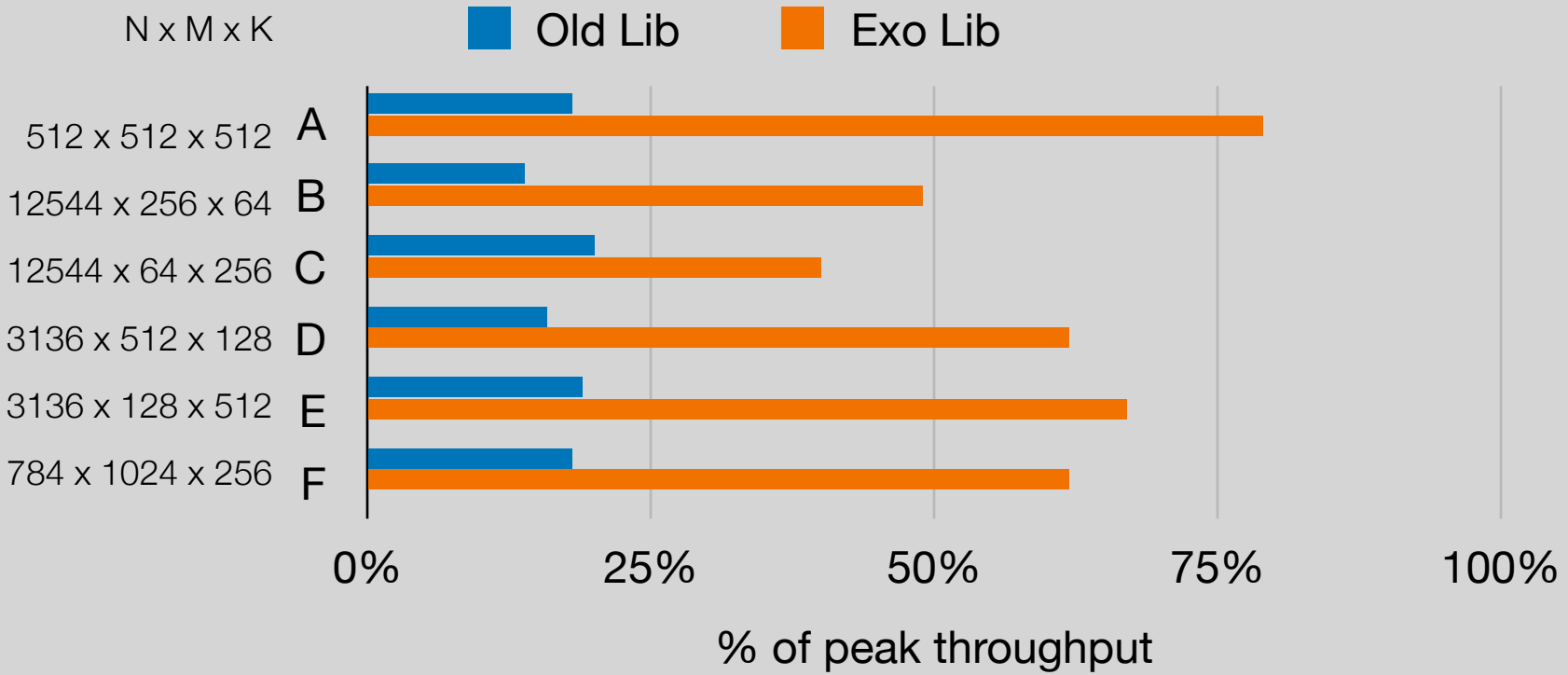
Results from 2021



GEMMINI MatMul

2.3x - 3.7x faster than original HPC Library

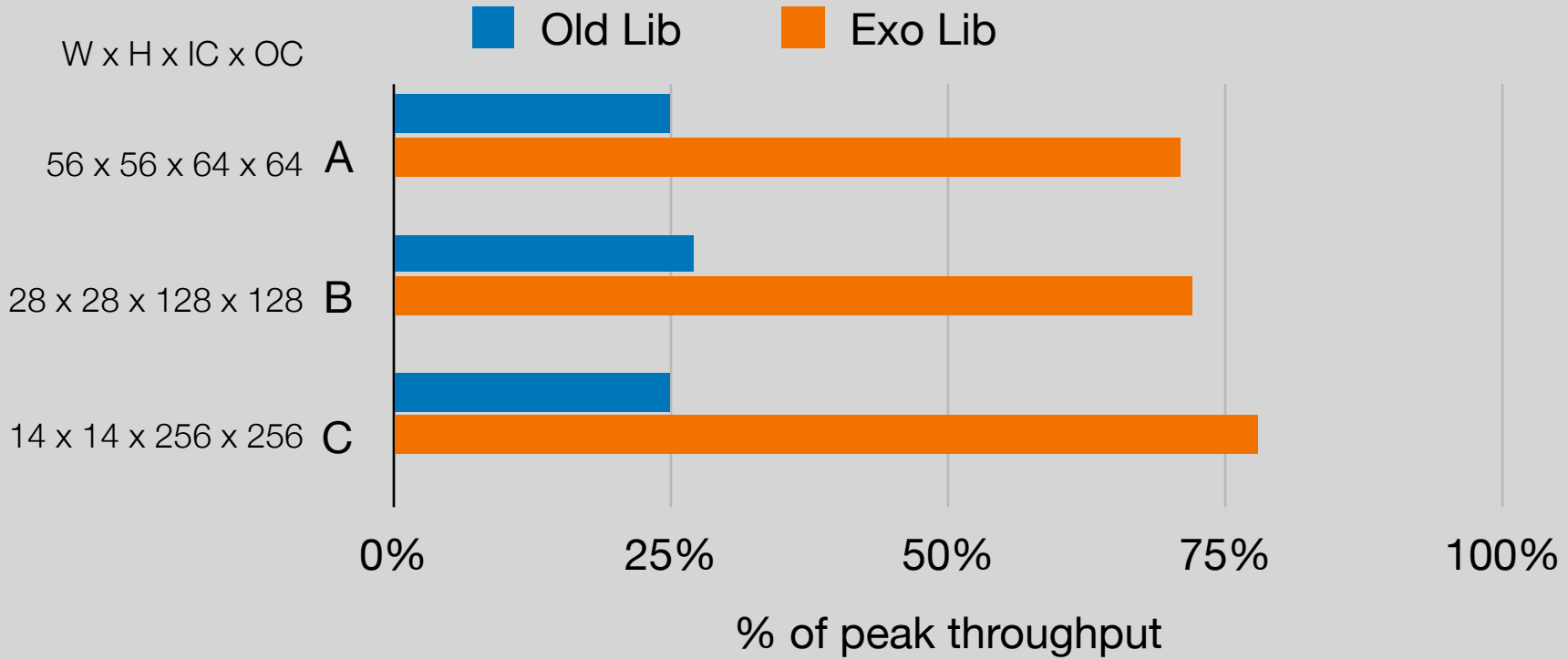
Results from 2021



GEMMINI Conv

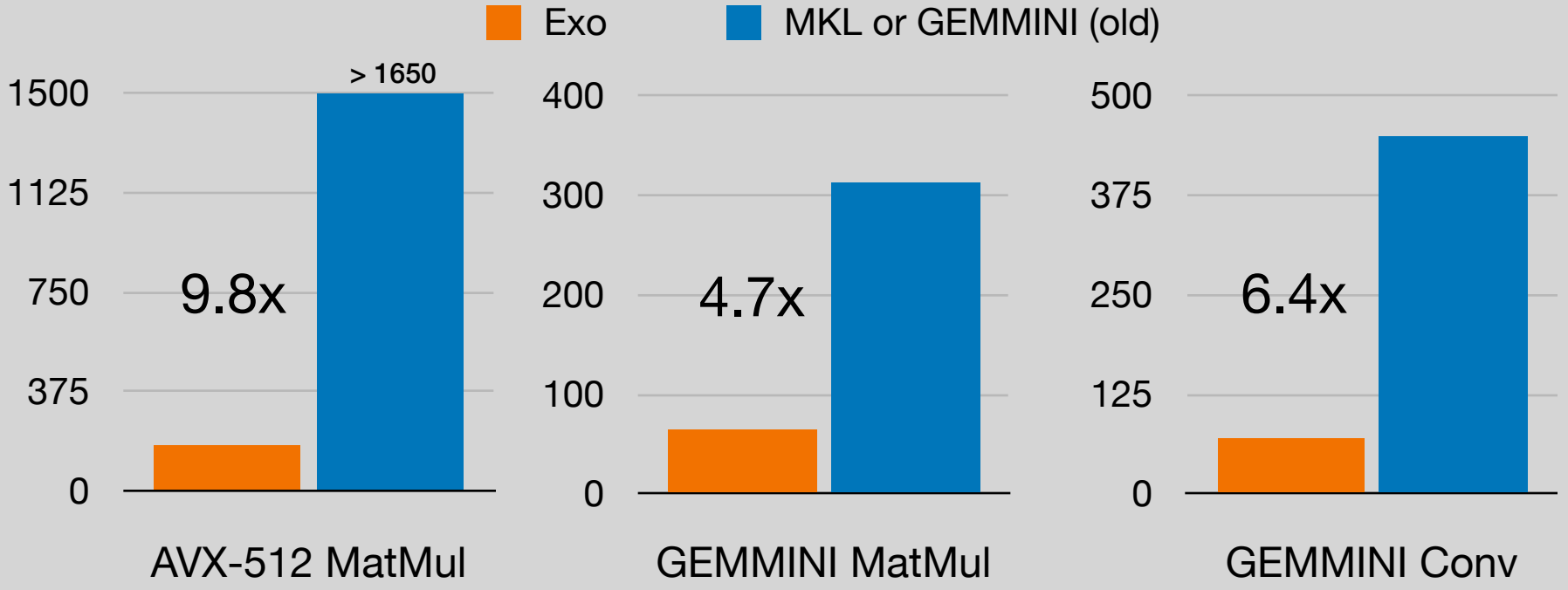
Similar Story for Convolution Layers

Results from 2021



Exo vs. Existing - Lines of Code

Results from 2021



Some Important Limitations

`32*j 4*k + 8`

Linear/Affine Indexing Expression

`i*j + k`

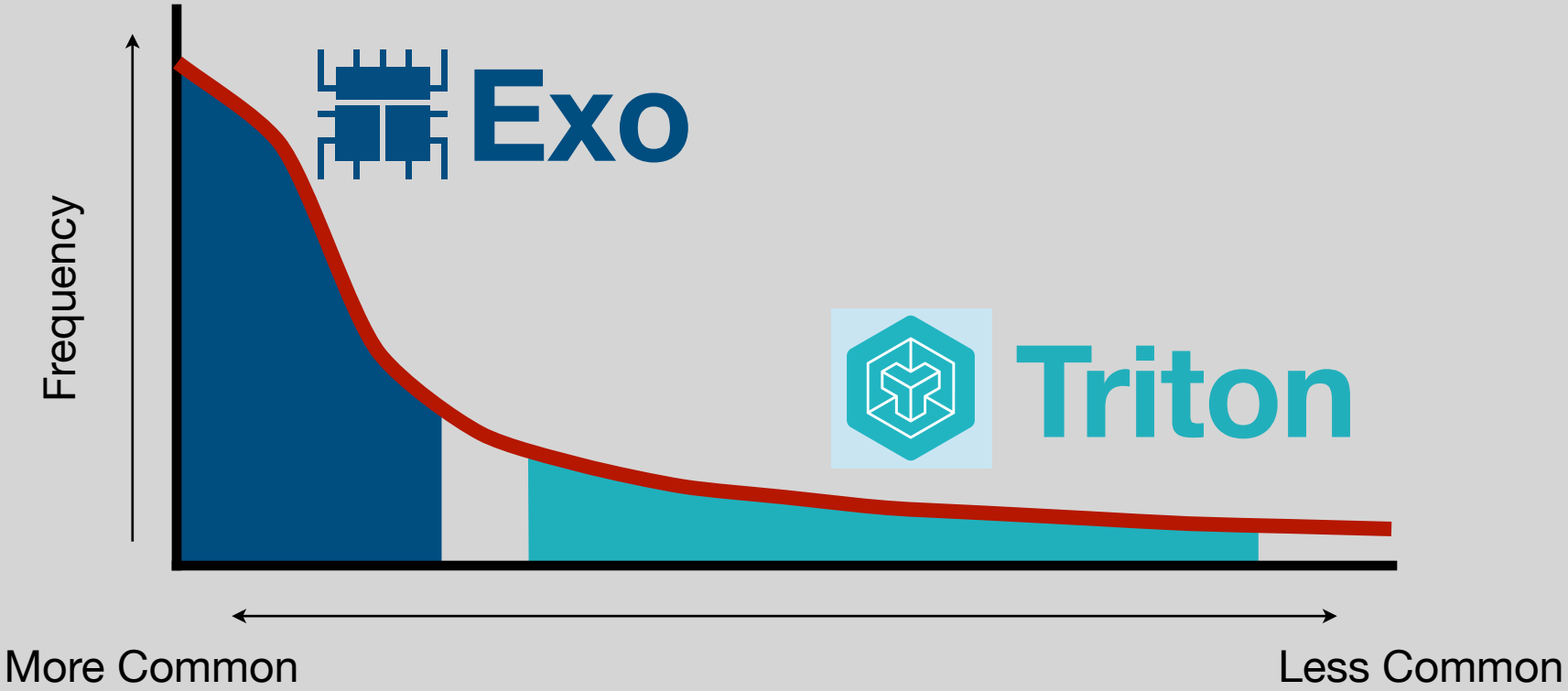
Non-Linear Indexing

Currently Unsupported

UNIVERSITY of WASHINGTON

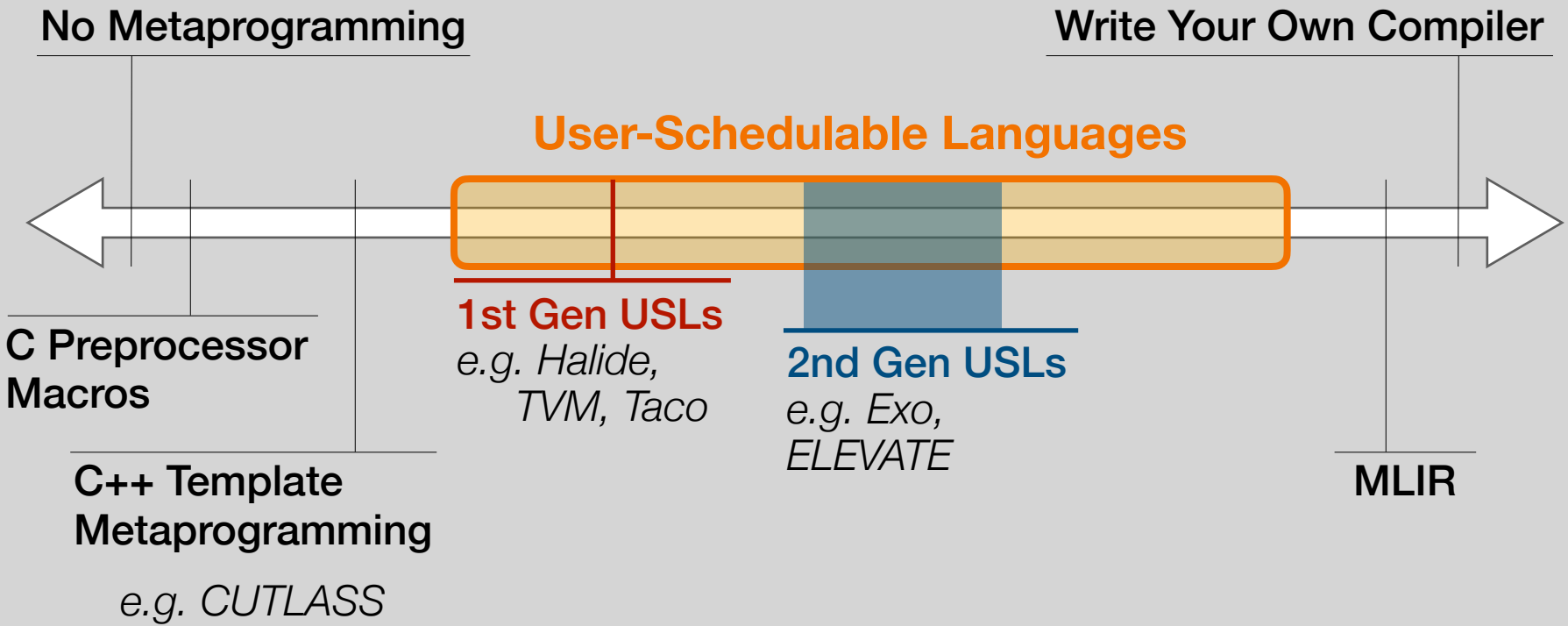
Massachusetts Institute of Technology

Distribution of Kernel Workload

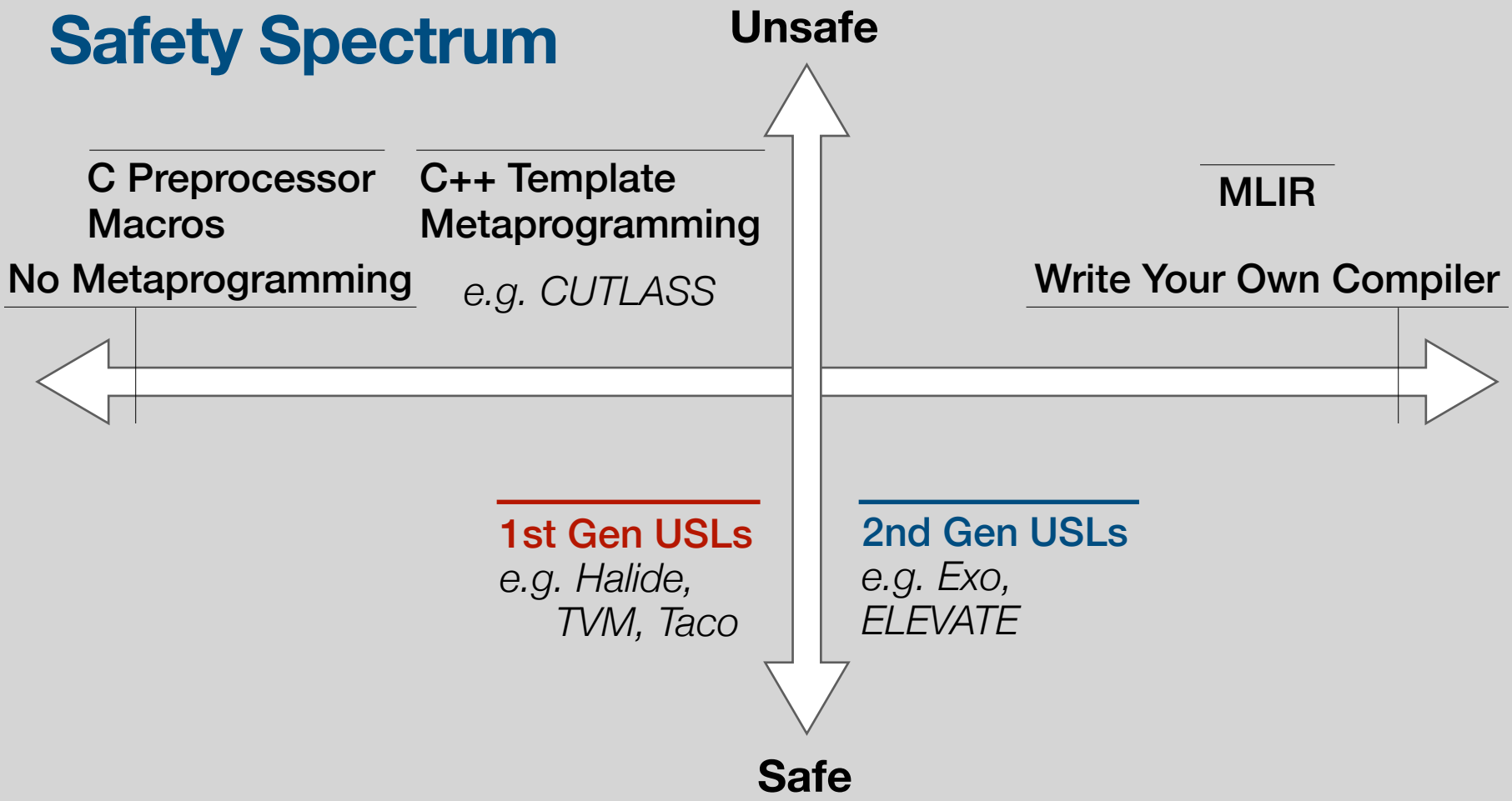


Metaprogramming Spectrum

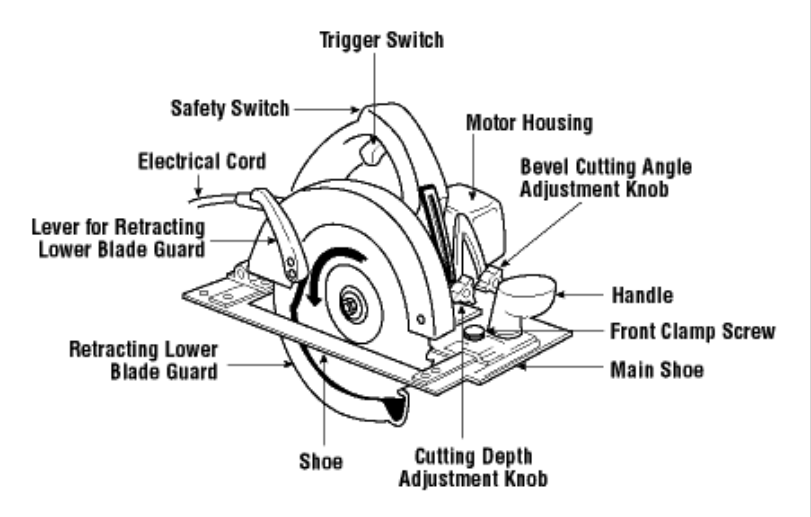
Metaprogramming Spectrum

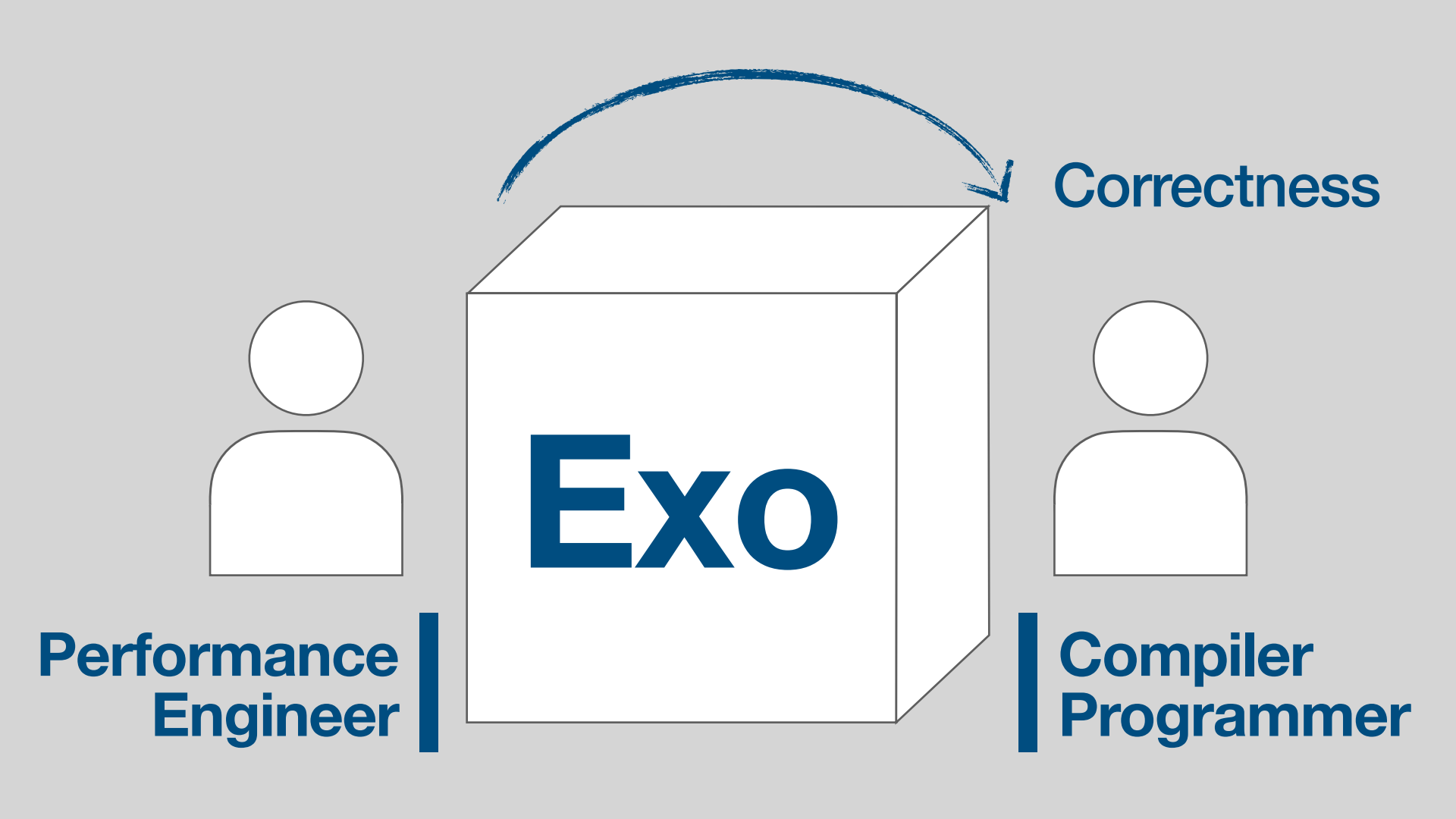


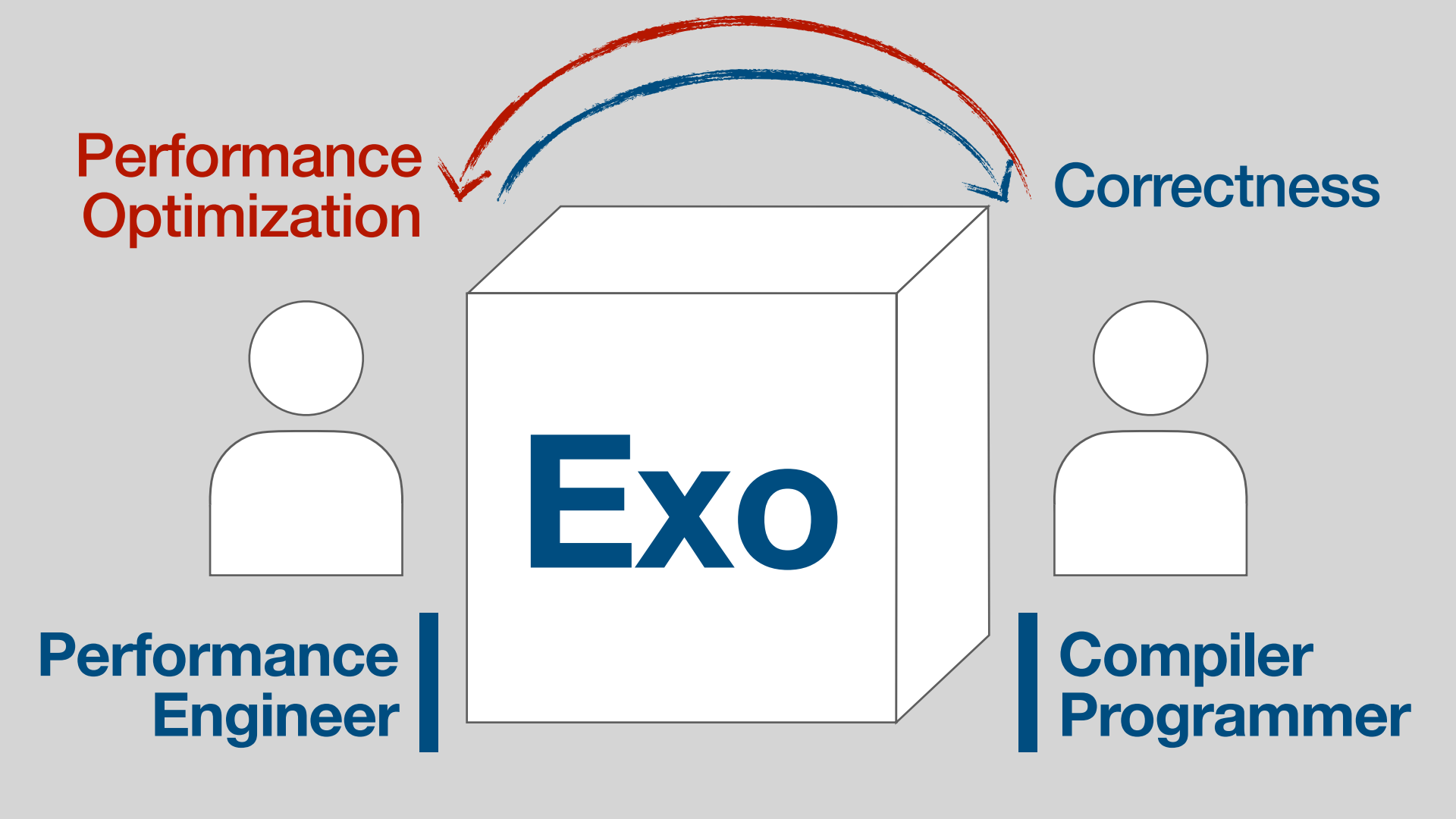
Safety Spectrum

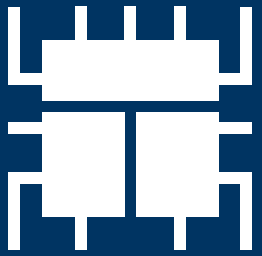


Power Tools have Safety Features




Correctness
Exo
Performance Engineer
Compiler Programmer


Performance
Optimization
Correctness
Exo
Performance
Engineer
Compiler
Programmer



Exo

Tutorials & Documentation are…