Tensor Comprehensions

Tensor Comprehensions

- Published in 2018
- Researcher wants to develop a novel type of layer or network architecture
- Must develop a customer operator
  - High engineering cost
  - Performance penalty
- Novel domain-specific flow capable of generating highly-optimized kernels for tensor expressions (for GPU)
  - For new operators, we want to find an efficient implementation easily, without thinking about the hardware

Contributions

- Domain-specific language “Tensor Comprehensions (TC)”
  - Syntax is both concise and expressive
  - Semantics allows for efficient memory management and mapping to complex parallel platforms
- Specialize a polyhedral intermediate representation and compilation algorithm to DL, provide a dedicated autotuner
- Enables rapid prototyping of new operators for researchers
- With comparable performance than manual tuning

Tensor Comprehensions: A Notation

- Three ideas from Einstein notation
- Index variables are defined implicitly by using them. Range is inferred.
- Indices that appear only on the right of an expression are assumed to be reduction dimensions.
- Evaluation order of points in the iteration space does not affect the output.

```
def mv(float(M,K) A, float(K) x) → (C) {
  C(i)   = 0
  C(i) += A(i,k) * x(k)
}
```

```
tensor C({M}).zero(); // 0-filled single-dim tensor
parallel for (int i = 0; i < M; i++)
  reduction for (int k = 0; k < K; k++)
    C(i) += A(i,k) * x(k);
```

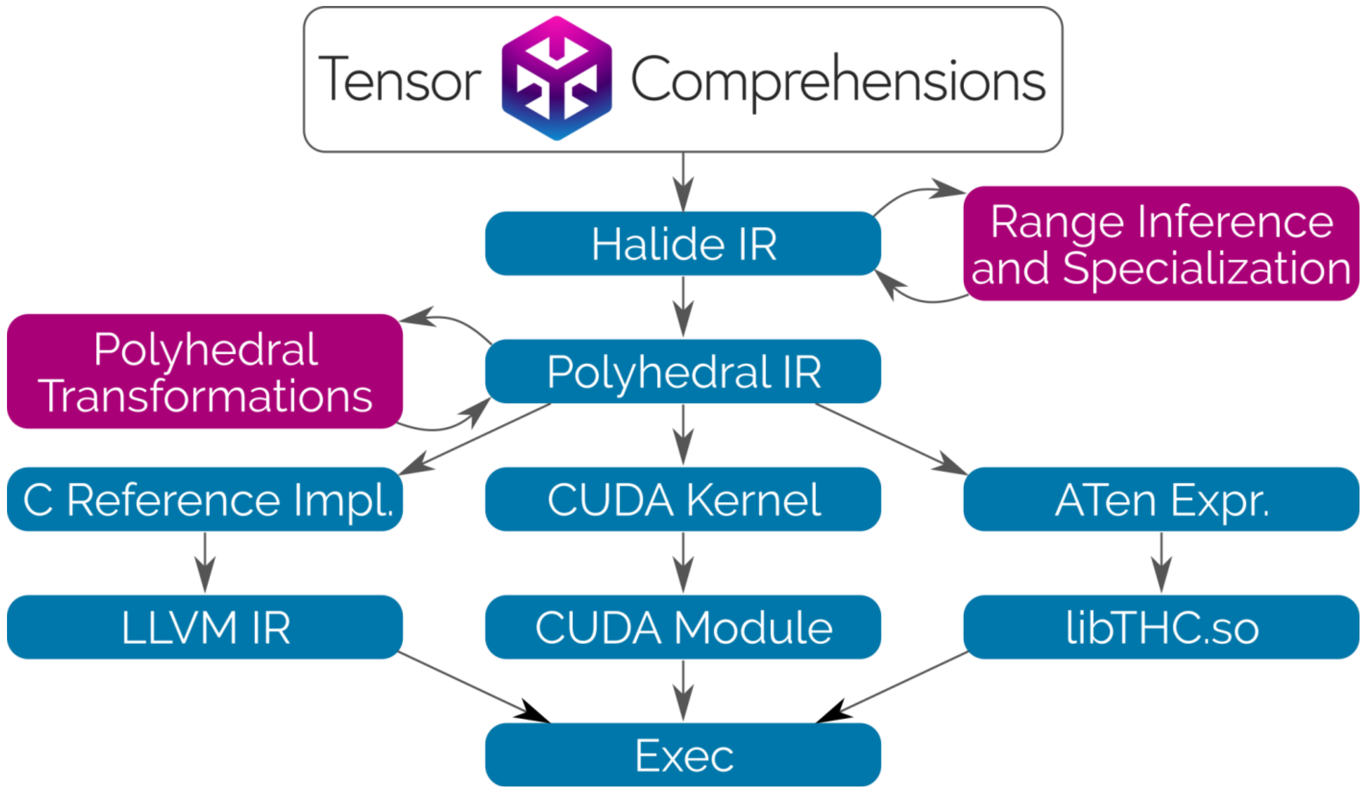
Why TC?

- TC DSL is extremely simple
- Resembles the ***whiteboard mathematical model*** of a deep neural network
- Makes it easy to reason about, communicate, and to manually alter the computations and storage/computation tradeoffs

```
def sgemm(float a, float b,
          float(N,M) A, float(M,K) B) → (C) {
  C(i,j)  = b * C(i,j)           # initialization
  C(i,j) += a * A(i,k) * B(k,j)  # accumulation
}
```

Figure 1: Tensor Comprehension for the sgemm BLAS

TC Flow



Tensor Comprehensions
Halide IR
Range Inference and Specialization
Polyhedral Transformations
Polyhedral IR
C Reference Impl.
CUDA Kernel
ATen Expr.
LLVM IR
CUDA Module
libTHC.so
Exec

TC: Halide + Polyhedral

- Tensor Comprehensions use Halide and Polyhedral Compilation techniques
  - Automatically synthesize CUDA kernels
  - With delegated memory management and synchronization.

Optimization

```
def sgemm(float a, float b,
          float(N,M) A, float(M,K) B) → (C) {
  C(i,j)  = b * C(i,j)              # initialization
  C(i,j) += a * A(i,k) * B(k,j)     # accumulation
}
```

Figure 1: Tensor Comprehension for the sgemm BLAS

$$C_{i,j} = \sum_{k=1}^{N} A_{i,k} \cdot B_{k,j}, \quad \forall i, j \in 1, N$$

$$\begin{array}{l}
\text{Domain} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & | \, 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathtt{T}(i,j,k) & | \, 0 \le i < N \\ & \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array} \right. \\
\quad \text{Sequence} \\
\quad\quad \text{Filter}\{\mathtt{S}(i,j)\} \\
\quad\quad\quad \text{Band}\{\mathtt{S}(i,j) \to (i,j)\} \\
\quad\quad \text{Filter}\{\mathtt{T}(i,j,k)\} \\
\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k) \to (i,j,k)\}
\end{array}$$

(a) canonical sgemm

$$\begin{array}{l}
\text{Domain} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & | \, 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathtt{T}(i,j,k) & | \, 0 \le i < N \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array} \right. \\
\quad \text{Band} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & \to (i,j)\} \\ \{\mathtt{T}(i,j,k) & \to (i,j)\} \end{array} \right. \\
\quad\quad \text{Sequence} \\
\quad\quad\quad \text{Filter}\{\mathtt{S}(i,j)\} \\
\quad\quad\quad \text{Filter}\{\mathtt{T}(i,j,k)\} \\
\quad\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k) \to (k)\}
\end{array}$$

(b) fused

$$\begin{array}{l}
\text{Domain} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & | \, 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathtt{T}(i,j,k) & | \, 0 \le i < N \\ & \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array} \right. \\
\quad \text{Band} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \\ \{\mathtt{T}(i,j,k) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \end{array} \right. \\
\quad\quad \text{Band} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & \to (i \bmod 32, j \bmod 32)\} \\ \{\mathtt{T}(i,j,k) & \to (i \bmod 32, j \bmod 32)\} \end{array} \right. \\
\quad\quad\quad \text{Sequence} \\
\quad\quad\quad\quad \text{Filter}\{\mathtt{S}(i,j)\} \\
\quad\quad\quad\quad \text{Filter}\{\mathtt{T}(i,j,k)\} \\
\quad\quad\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k) \to (k)\}
\end{array}$$

(c) fused and tiled

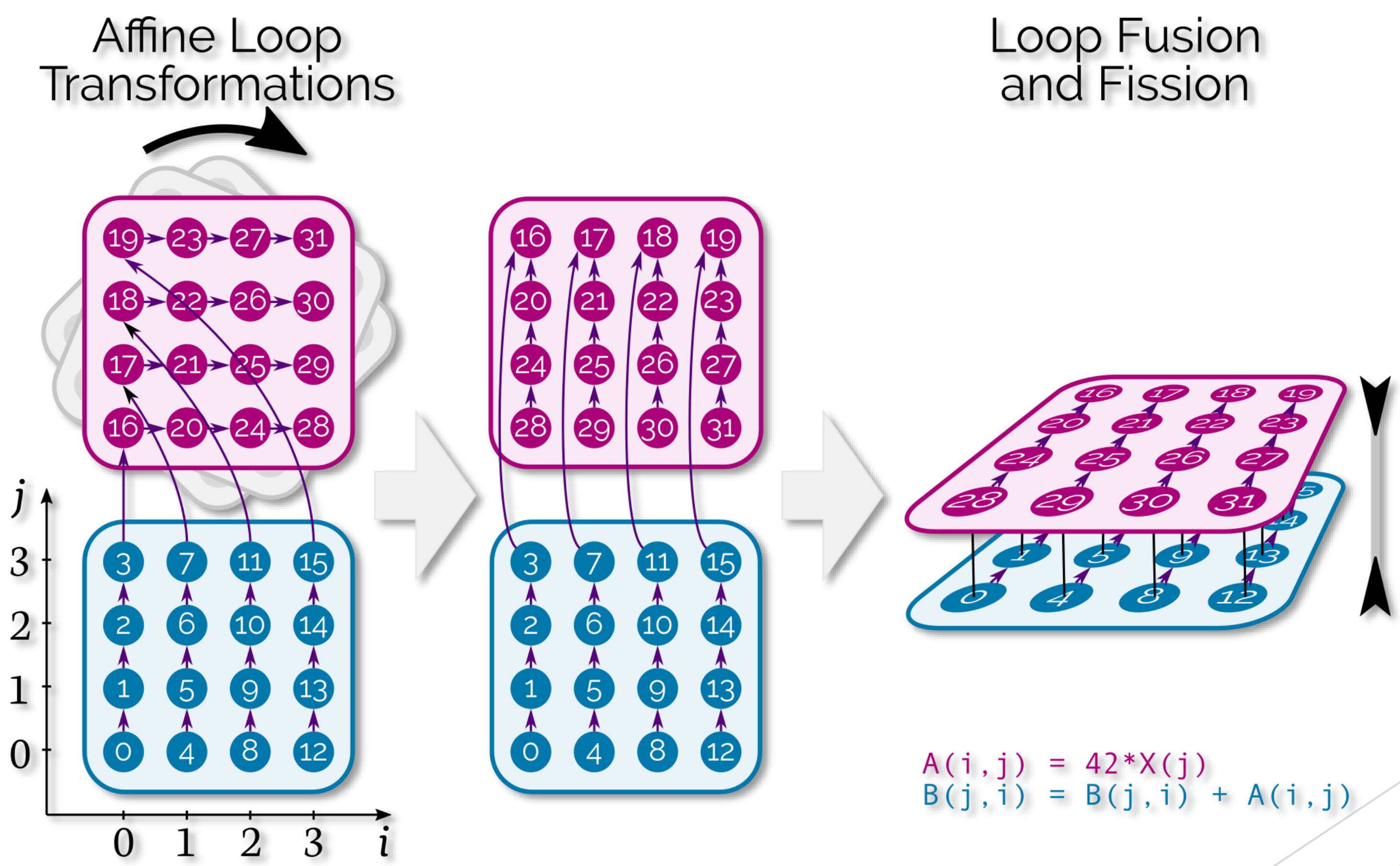
$$\begin{array}{l}
\text{Domain} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & | \, 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathtt{T}(i,j,k) & | \, 0 \le i < N \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array} \right. \\
\quad \text{Band} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \\ \{\mathtt{T}(i,j,k) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \end{array} \right. \\
\quad\quad \text{Sequence} \\
\quad\quad\quad \text{Filter}\{\mathtt{S}(i,j)\} \\
\quad\quad\quad\quad \text{Band}\{\mathtt{S}(i,j) \to (i \bmod 32, j \bmod 32)\} \\
\quad\quad\quad \text{Filter}\{\mathtt{T}(i,j,k)\} \\
\quad\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k) \to (32\lfloor k/32 \rfloor)\} \\
\quad\quad\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k) \to (k \bmod 32)\} \\
\quad\quad\quad\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k0 \to (i \bmod 32, j \bmod 32)\}
\end{array}$$

(d) fused, tiled and sunk

$$\begin{array}{l}
\text{Domain} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & | \, 0 \le i < N \wedge 0 \le j < K\} \\ \{\mathtt{T}(i,j,k) & | \, 0 \le i < N \wedge 0 \le j < K \wedge 0 \le k < M\} \end{array} \right. \\
\quad \text{Context}\{0 \le b_x, b_y < 32 \wedge 0 \le t_x, t_y < 16\} \\
\quad\quad \text{Filter} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & | \, i - 32b_x - 31 \le 32 \times 16\lfloor i/32/16 \rfloor \le i - 32b_x \wedge \\ & \;\; j - 32b_y - 31 \le 32 \times 16\lfloor j/32/16 \rfloor \le j - 32b_y\} \\ \{\mathtt{T}(i,j,k) & | \, i - 32b_x - 31 \le 32 \times 16\lfloor i/32/16 \rfloor \le i - 32b_x \wedge \\ & \;\; j - 32b_y - 31 \le 32 \times 16\lfloor j/32/16 \rfloor \le j - 32b_y\} \end{array} \right. \\
\quad\quad\quad \text{Band} \left[\begin{array}{ll} \{\mathtt{S}(i,j) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \\ \{\mathtt{T}(i,j,k) & \to (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \end{array} \right. \\
\quad\quad\quad\quad \text{Sequence} \\
\quad\quad\quad\quad\quad \text{Filter}\{\mathtt{S}(i,j)\} \\
\quad\quad\quad\quad\quad\quad \text{Filter} \;\; \{\mathtt{S}(i,j) \,|\, (t_x - i) = 0 \bmod 16 \wedge (t_y - j) = 0 \bmod 16\} \\
\quad\quad\quad\quad\quad\quad\quad \text{Band}\{\mathtt{S}(i,j) \to (i \bmod 32, j \bmod 32)\} \\
\quad\quad\quad\quad\quad \text{Filter}\{\mathtt{T}(i,j,k)\} \\
\quad\quad\quad\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k) \to (32\lfloor k/32 \rfloor)\} \\
\quad\quad\quad\quad\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k) \to (k \bmod 32)\} \\
\quad\quad\quad\quad\quad\quad\quad\quad \text{Filter} \;\; \begin{array}{l} \{\mathtt{T}(i,j,k) \,|\, (t_x - i) = 0 \bmod 16 \wedge \\ \quad (t_y - j) = 0 \bmod 16\} \end{array} \\
\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{Band}\{\mathtt{T}(i,j,k) \to (i \bmod 32, j \bmod 32)\}
\end{array}$$

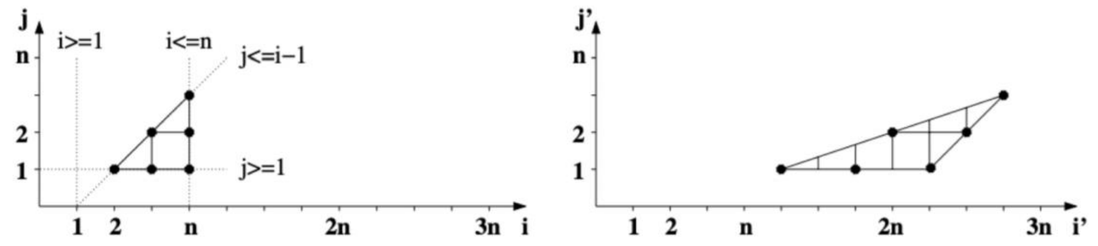
(e) fused, tiled, sunk and mapped

Optimization



TC: Polyhedral Transformaion + Mapping

- Transform for parallelism and data locality
- Map GPU compute and memory resources to the transformed program



$$\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{pmatrix} i \\ j \end{pmatrix} \geq \begin{pmatrix} 1 \\ -n \\ 1 \\ 1 \end{pmatrix}$$

(a) original polyhedron $A\vec{x} \geq \vec{c}$

$$\begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \\ 0 & 1 \\ 1/2 & -3/2 \end{bmatrix} \begin{pmatrix} i' \\ j' \end{pmatrix} \geq \begin{pmatrix} 1 \\ -n \\ 1 \\ 1 \end{pmatrix}$$

(b) usual transformation $(AT^{-1})\vec{y} \geq \vec{c}$

Domain $\left[\begin{array}{l} \{\mathtt{S}(i,j) \mid 0 \leq i < N \wedge 0 \leq j < K\} \\ \{\mathtt{T}(i,j,k) \mid 0 \leq i < N \wedge 0 \leq j < K \wedge 0 \leq k < M\} \end{array}\right.$

Context $\{0 \leq b_x, b_y < 32 \wedge 0 \leq t_x, t_y < 16\}$

Filter $\left[\begin{array}{l} \{\mathtt{S}(i,j) \mid i - 32b_x - 31 \leq 32 \times 16\lfloor i/32/16 \rfloor \leq i - 32b_x \wedge \\ \quad j - 32b_y - 31 \leq 32 \times 16\lfloor j/32/16 \rfloor \leq j - 32b_y\} \\ \{\mathtt{T}(i,j,k) \mid i - 32b_x - 31 \leq 32 \times 16\lfloor i/32/16 \rfloor \leq i - 32b_x \wedge \\ \quad j - 32b_y - 31 \leq 32 \times 16\lfloor j/32/16 \rfloor \leq j - 32b_y\} \end{array}\right.$

Band $\left[\begin{array}{l} \{\mathtt{S}(i,j) \rightarrow (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \\ \{\mathtt{T}(i,j,k) \rightarrow (32\lfloor i/32 \rfloor, 32\lfloor j/32 \rfloor)\} \end{array}\right.$

Sequence

Filter$\{\mathtt{S}(i,j)\}$

Filter $\{\mathtt{S}(i,j) \mid (t_x - i) = 0 \bmod 16 \wedge (t_y - j) = 0 \bmod 16\}$

Band$\{\mathtt{S}(i,j) \rightarrow (i \bmod 32, j \bmod 32)\}$

Filter$\{\mathtt{T}(i,j,k)\}$

Band$\{\mathtt{T}(i,j,k) \rightarrow (32\lfloor k/32 \rfloor)\}$

Band$\{\mathtt{T}(i,j,k) \rightarrow (k \bmod 32)\}$

Filter $\{\mathtt{T}(i,j,k) \mid (t_x - i) = 0 \bmod 16 \wedge (t_y - j) = 0 \bmod 16\}$

Band$\{\mathtt{T}(i,j,k) \rightarrow (i \bmod 32, j \bmod 32)\}$

(e) fused, tiled, sunk and mapped

Autotuning

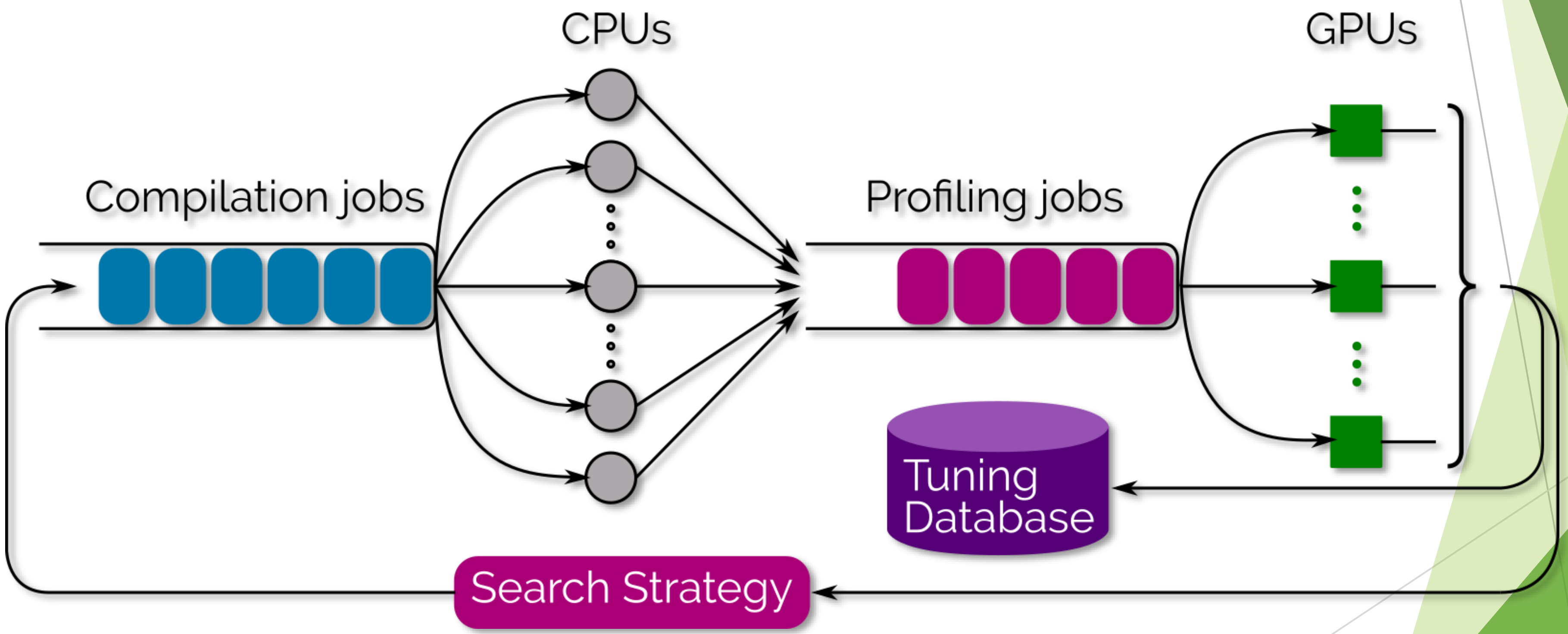
- Autotuner interacts with the rest of the environment through the compilation cache: best versions are stored for later use.
- Compilation cache stores the generated CUDA or PTX code for a given TC
  - Generated code depends on the input shapes, the selected optimization options, constraints induced by the target GPU architecture
- Autotuner runs for a prescribed amount of time, updating the cache with better versions along the way using genetic algorithm
  - Three parents are selected probabilistically based on their fitness, the higher the fitness the higher the selection chance
  - Each "gene", which corresponds to one tuning parameter, of the new candidate is randomly selected from the parents.

Autotuning

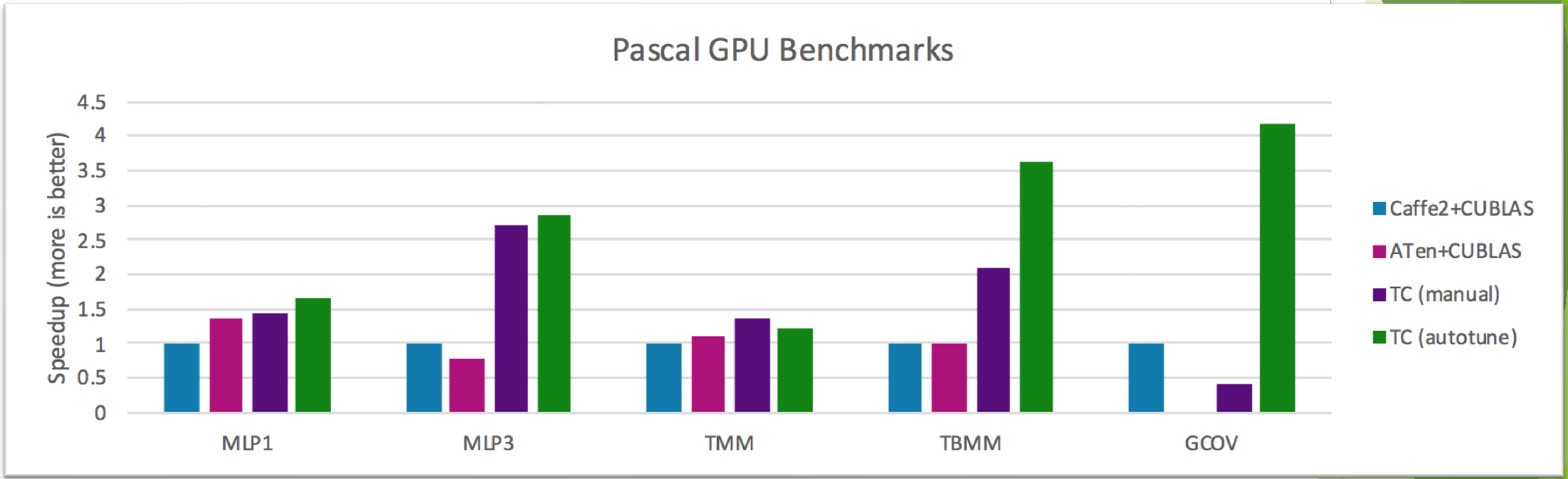
```
def avgpool(float(B, C, H, W) input) -> (output) {{
    output(b, c, h, w) += input(b, c, h * {sH} + kh, w * {sW} + kw)
                          where kh in 0:{kH}, kw in 0:{kW}
}}
```

Tensor Comprehension for 2D Average Pooling

Multithreaded autotuning pipeline



TC Performance



```python
import tensor_comprehensions as tc
import torch
lang = """
def tensordot(float(N, C1, C2, H, W) I0, float(N, C2, C3, H, W) I1) -> (O) {
    O(n, c1, c3, h, w) +=! I0(n, c1, c2, h, w) * I1(n, c2, c3, h, w)
}
"""
N, C1, C2, C3, H, W = 32, 512, 8, 2, 28, 28
tensordot = tc.define(lang, name="tensordot")
I0, I1 = torch.randn(N, C1, C2, H, W).cuda(), torch.randn(N, C2, C3, H, W).cuda()
best_options = tensordot.autotune(I0, I1, cache=True)
out = tensordot(I0, I1, options=best_options)
```

```
ntv@devfair0172:~/TensorComprehensions$ ./build/test/test_autotuner --smoke_check=0 --tuner_threads=20 --tuner_gpus="0,1" --gtest_filter="*Dot*"
Note: Google Test filter = *Dot*
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from ATenCompilationUnitTest
[ RUN      ] ATenCompilationUnitTest.TensorDot

-----------------------------------------------------------
-------------------- KERNEL STATS ---------------------
-----------------      100 ITERATIONS    ----------------
-----------------------------------------------------------
Min: 4881us, p50: 4936us, p90: 5134us, p99: 5403us, Max: 5403us
-----------------------------------------------------------


-----------------------------------------------------------
----------------------  TOTAL STATS --------------------
-----------------      100 ITERATIONS    ----------------
-----------------------------------------------------------
Min: 4903us, p50: 4947us, p90: 5138us, p99: 5375us, Max: 5375us
-----------------------------------------------------------

Generation 0    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 4177/11678/621574
Generation 1    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2986/6414/20158
Generation 2    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2270/6193/14676
Generation 3    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2267/5608/11261
Generation 4    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2266/4817/11330
Generation 5    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2258/4632/11264
Generation 6    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2242/4634/10955
Generation 7    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2247/4488/10950
Generation 8    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2255/4333/10948
Generation 9    Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2253/4259/10370
Generation 10   Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2233/4261/10364
Generation 11   Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2233/4171/10371
Generation 12   Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 1968/4198/10365
Generation 13   Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 1959/3904/8670
Generation 14   Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 2084/3426/7846
Generation 15   Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 1959/3483/8929
Generation 16   Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 1959/3651/9465
Generation 17   Jobs(Compiled, GPU)/total  (100, 100)/100   (best/median/worst)us: 1812/3636/9392
Generation 18   Jobs(Compiled, GPU)/total  (100, 99)/100   (best/median/worst)us: 1784/3409/8828
```