End to End: Part 5

Earl Wong

Subjects

- Optics
- Sensor
- **ISP**
- GPU
- **NPU**

Different Naming Conventions in Mobile Space

- ANE (Apple) - Apple Neural Engine
- AI Engine (Qualcomm) - Hexagonal Tensor Processor
- APU (Mediatek) - AI Processing Unit
- NPU (Samsung) - Neural Processing Unit
- TPU (Google) - Tensor Processing Unit

Objective In Mobile Space

- Deliver fast, low latency ML inference under strict power and thermal constraints.

CPU vs GPU vs NPU

- CPU's are built for doing different types of work / general tasks = swiss army knife.
- The CPU runs the operating system.
- The CPU excels at control flow, because the CPU is designed for branching, prediction, and task management.
- The CPU excels at anything with unpredictable logic.

CPU vs GPU vs NPU

- GPU's originated, because of the need for faster graphics.
- In the process, GPU's sacrificed some flexibility, relative to their CPU counterparts.
- By construction, GPU's are graphic engines.
- By construction, GPU's exploit massive parallelism.
- Since image processing tasks are often a subset of graphics tasks, they can fit nicely into the GPU architecture.

CPU vs GPU vs NPU

- NPU's originated because of the need for fast neural network inference.
- By construction, NPU's are optimized for basic, fundamental neural network computations (multiply and accumulate (MAC)).
- To achieve high performance, nearly all flexibility is sacrificed.
- i.e. No branching, etc.
- Note: A CPU or a GPU could perform the same tasks as an NPU, but only slower and with more power usage.

Strengths; Weakness

- CPU - swiss army knife, control flow, high flexible; not optimized for GPU or NPU tasks
- GPU - massive parallelism; limited flexibility, high power
- NPU - dedicated, low power processor (MAC units) for fast, ML inference; minimal flexibility = one trick pony, but a very nice one

NPU

1) Dedicated and highly optimized ML operations

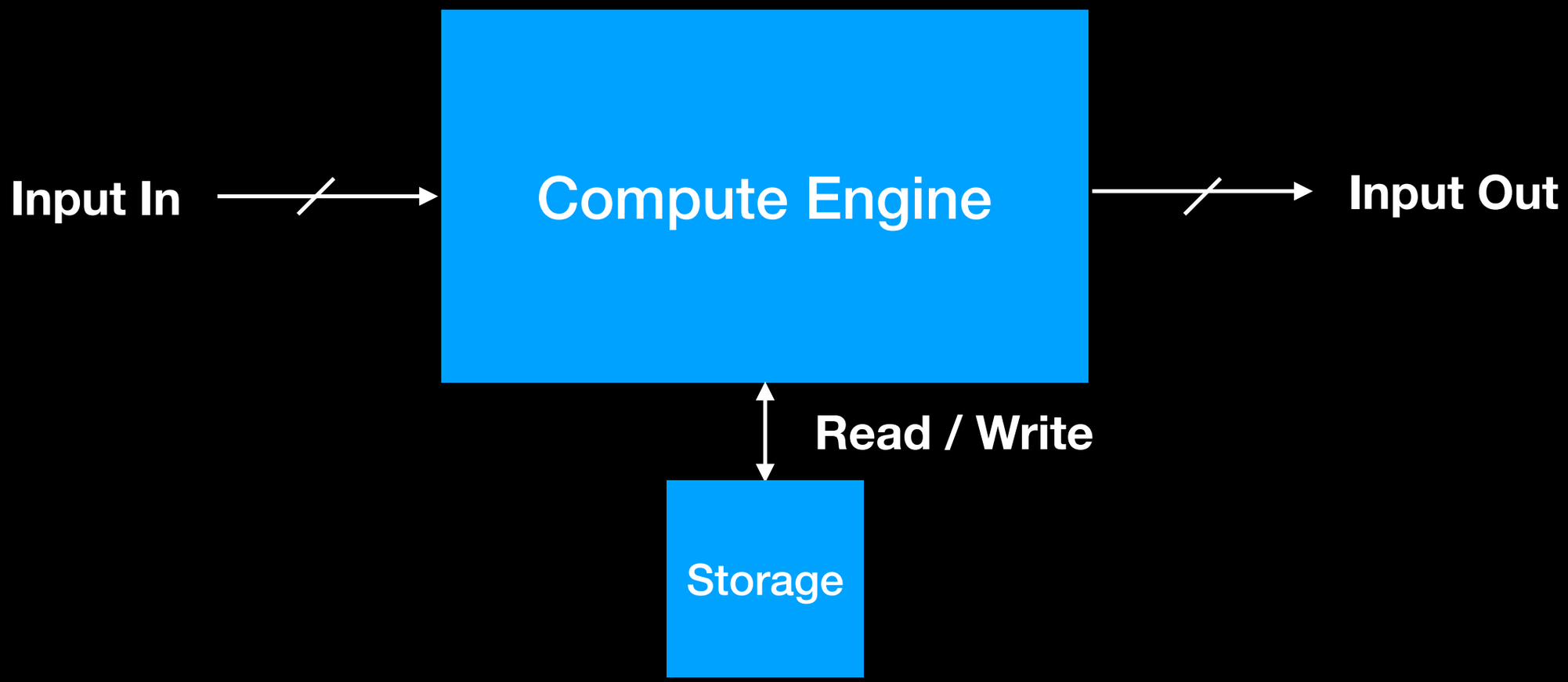
- Convolution (CNN’s)
- Matrix Multiplication (MLP’s)
- Activation (Non Linear Functionality)

2) Minimize DRAM fetch

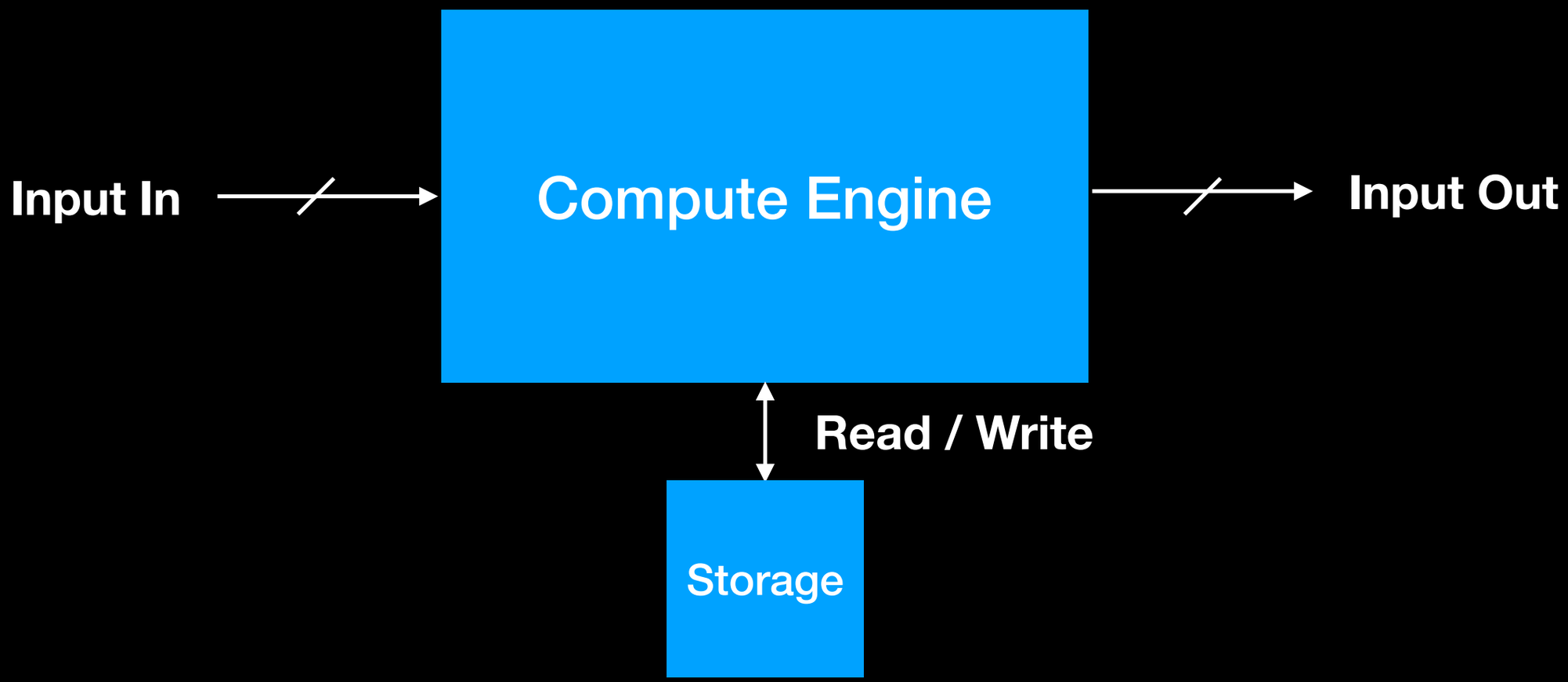
Data Fetch

- CPU’s and GPU’s utilize frequent data fetches from caches and DRAM.
- Caches misses and DRAM fetches are expensive, increasing power and bandwidth usage.
- NPU’s try to maximize data re-use, by streaming data through a fixed pipeline (=dataflow is fixed), minimizing external data fetches.
- Neural network weights are stored in the NPU SRAM.

High Level Evolution

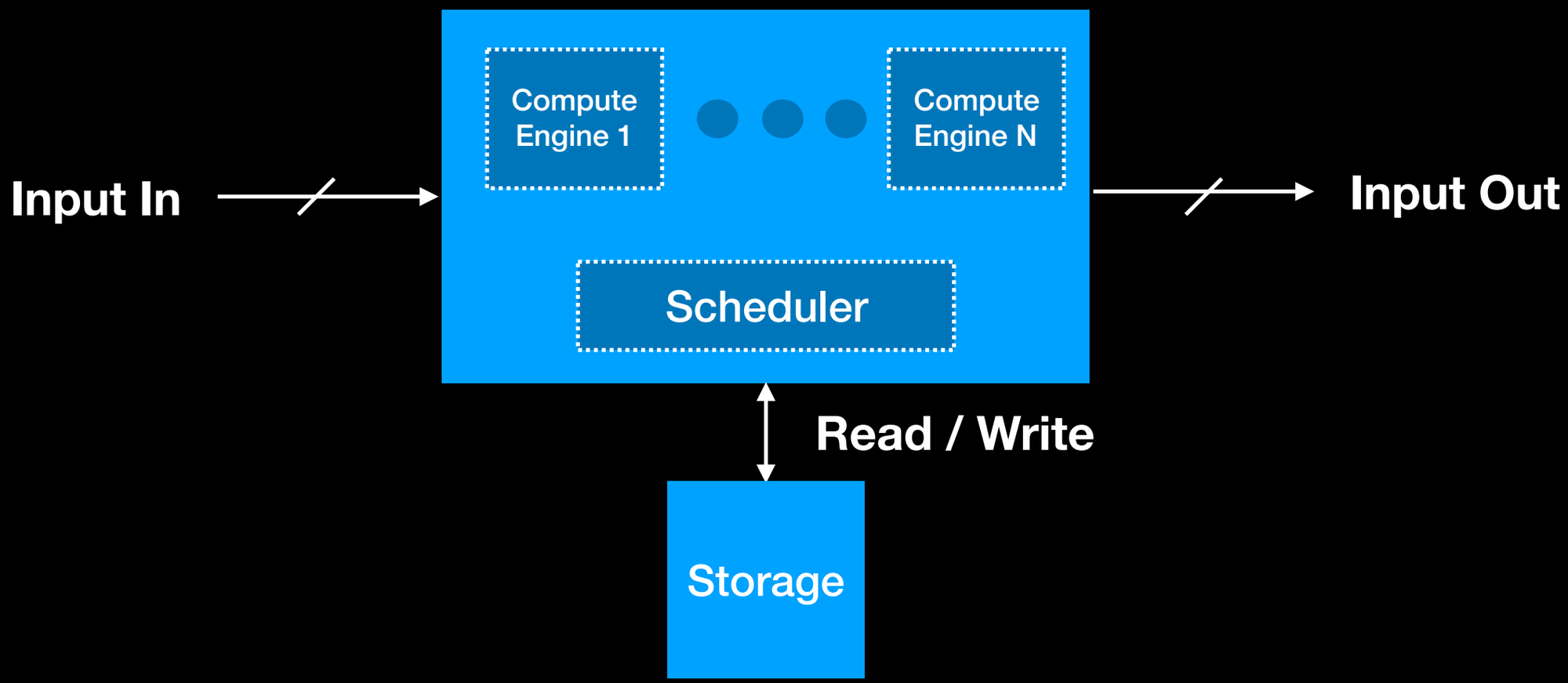


High Level Evolution

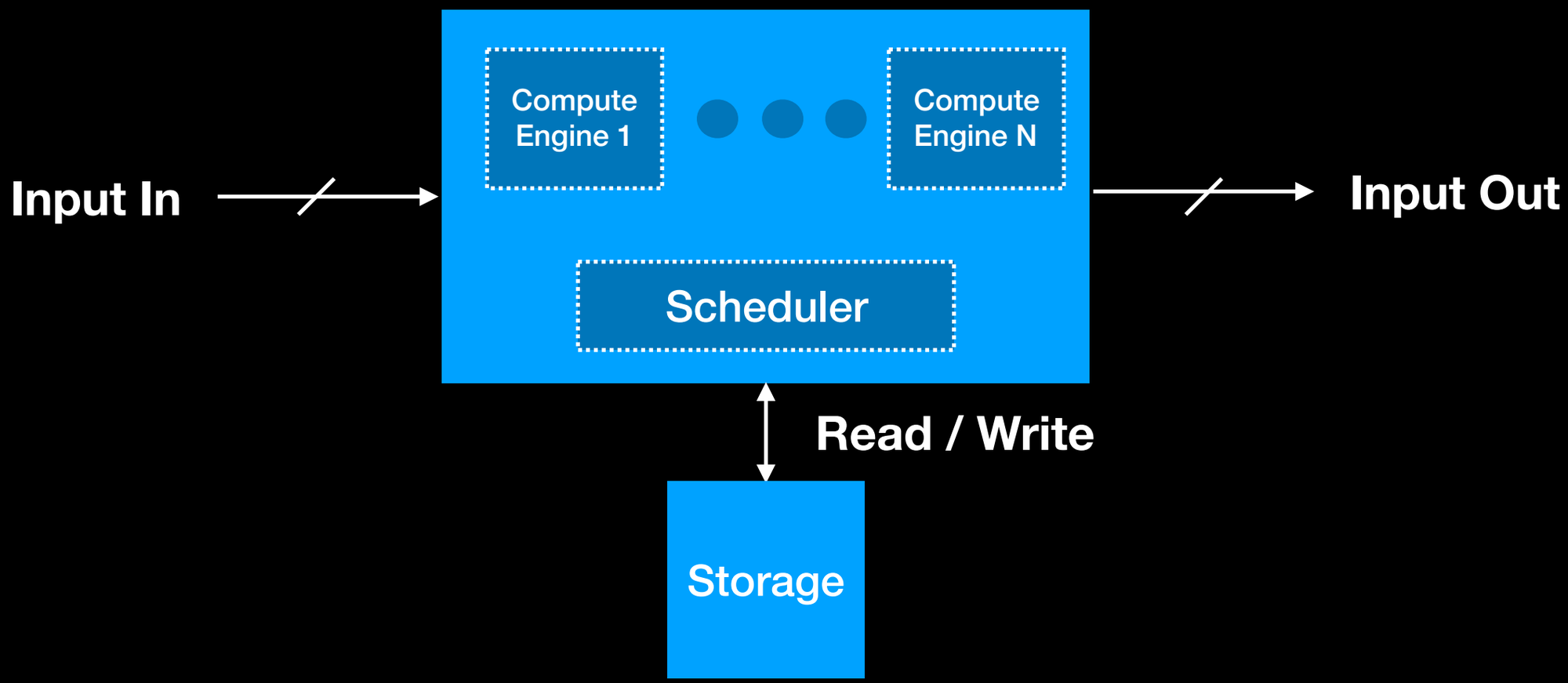


“I want more throughput!”

High Level Evolution

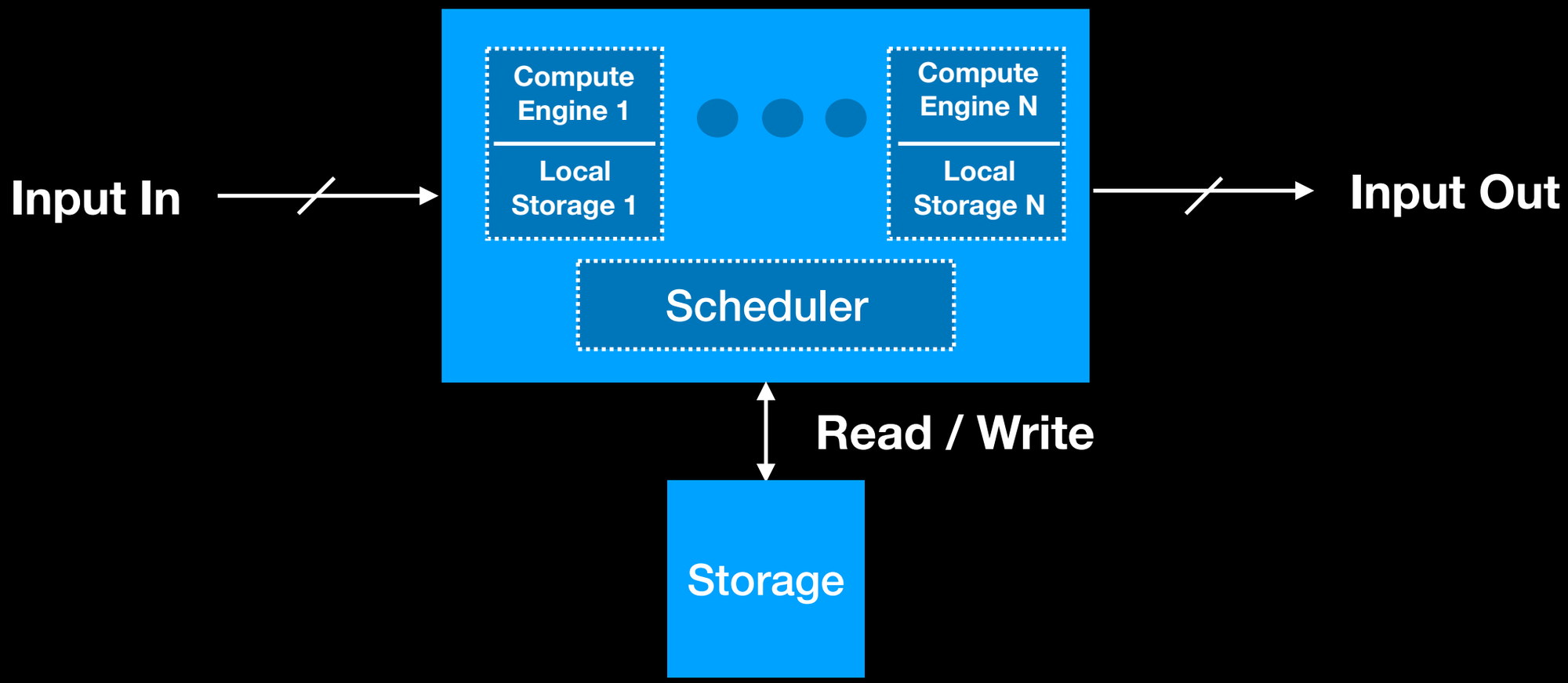


High Level Evolution

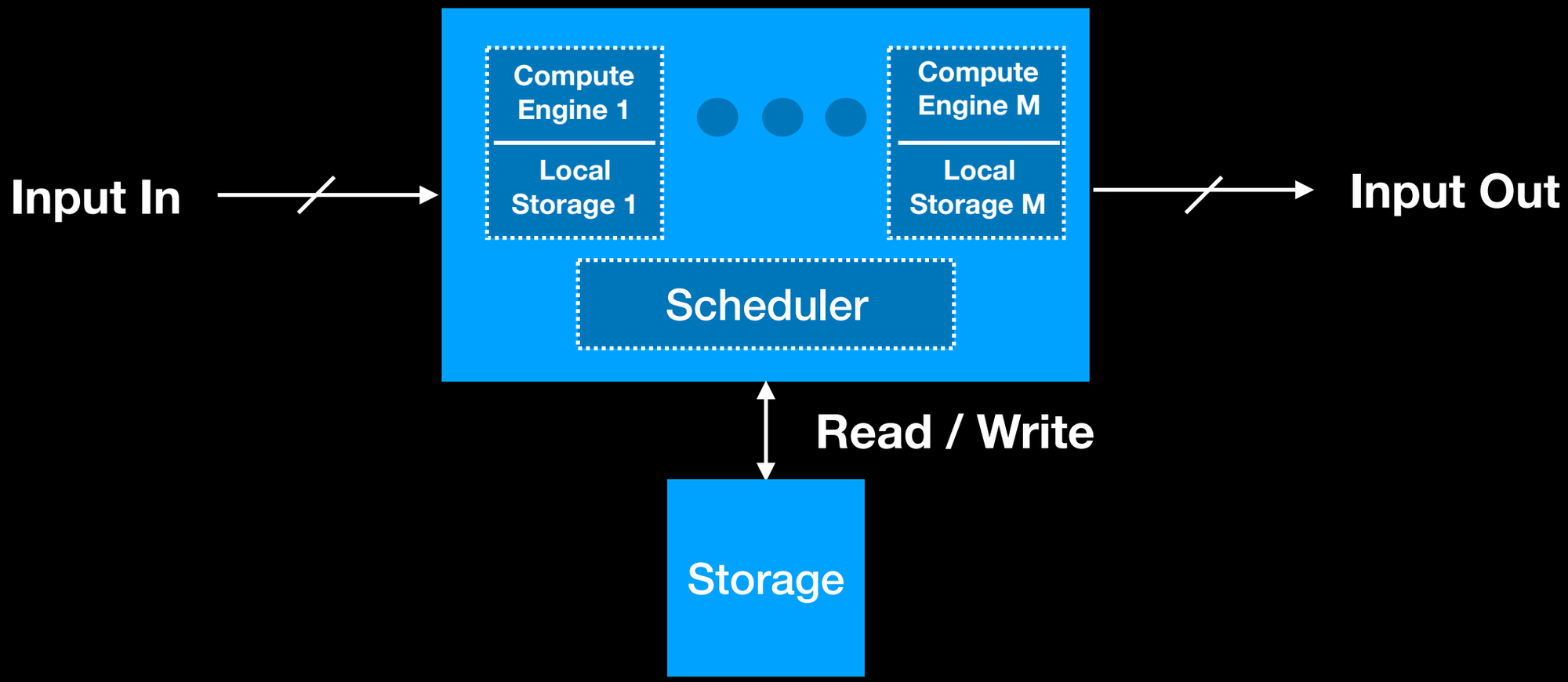


“The compute is under utilized, because we have to read / write from slow, external storage.”

High Level Evolution

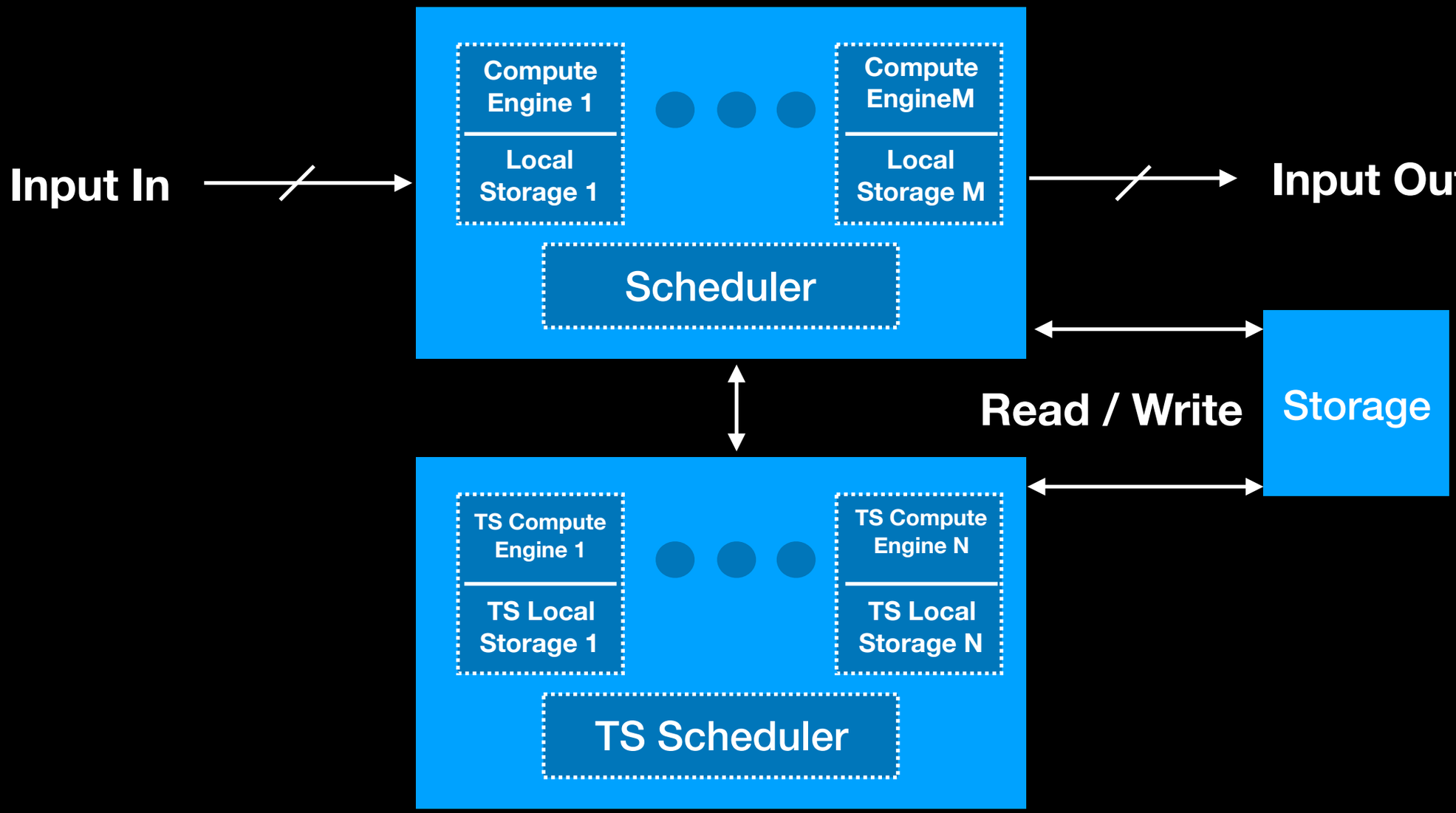


High Level Evolution

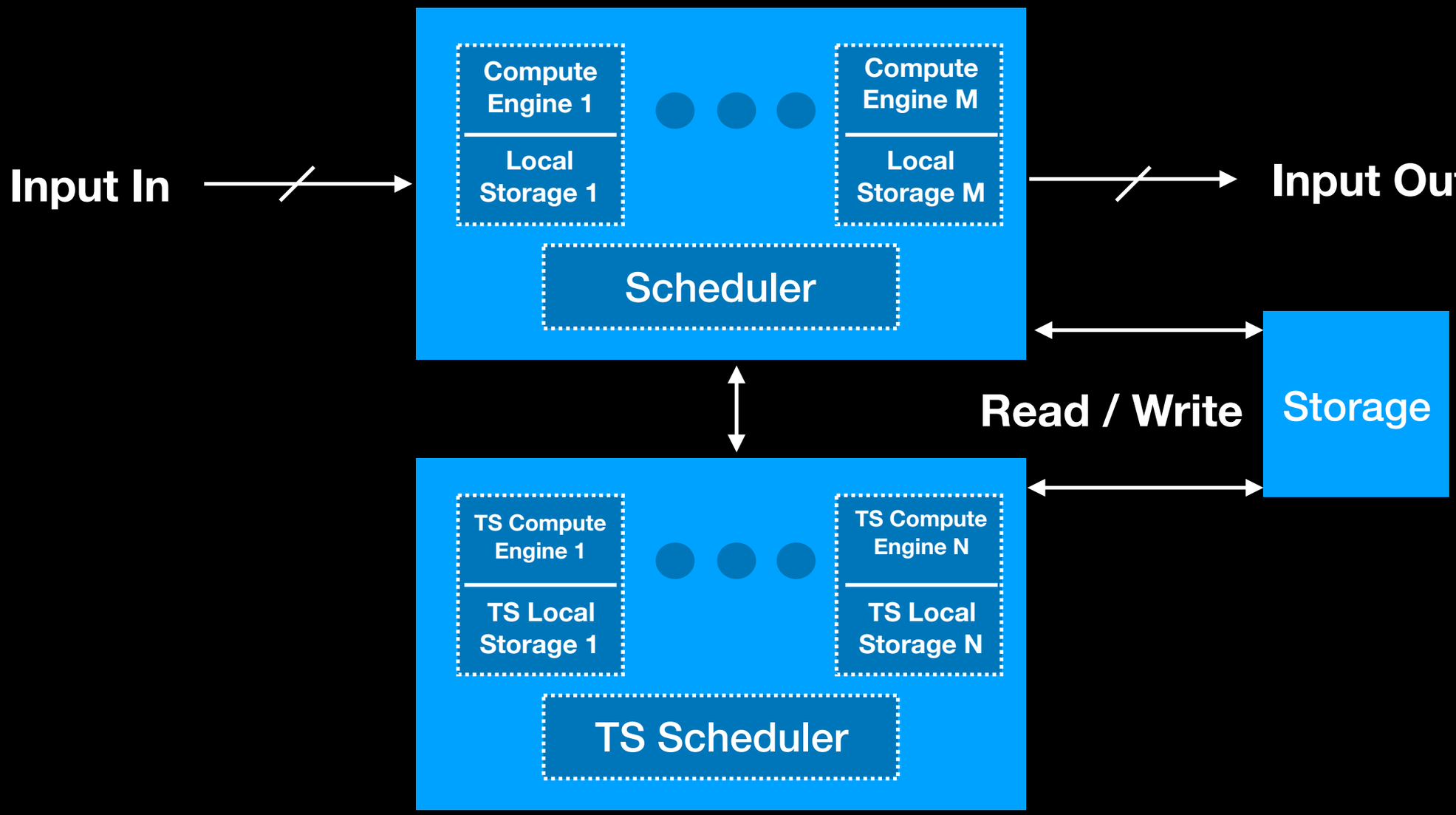


“The compute engine is inefficient / ill suited, for my task specific application.”

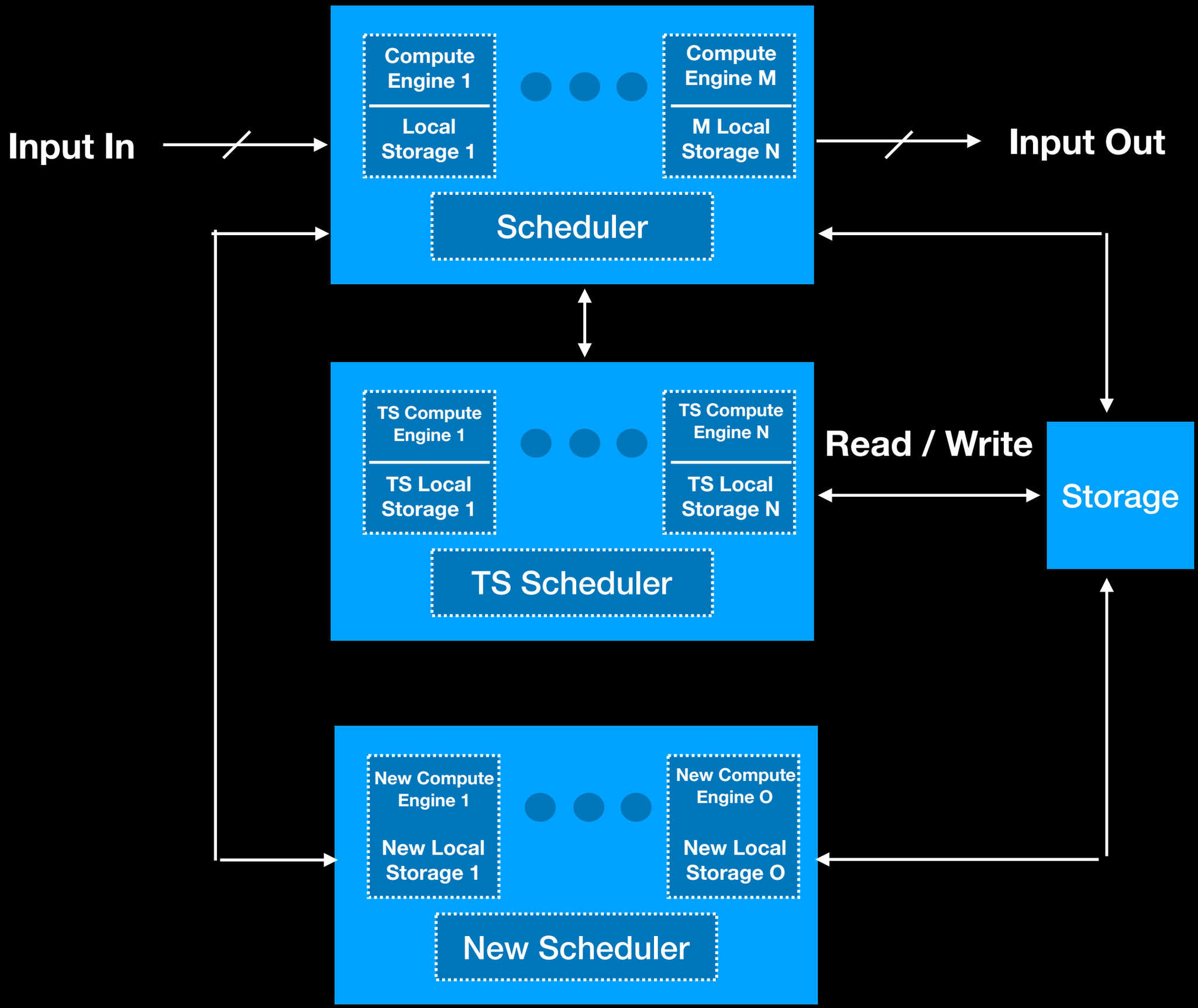
High Level Evolution

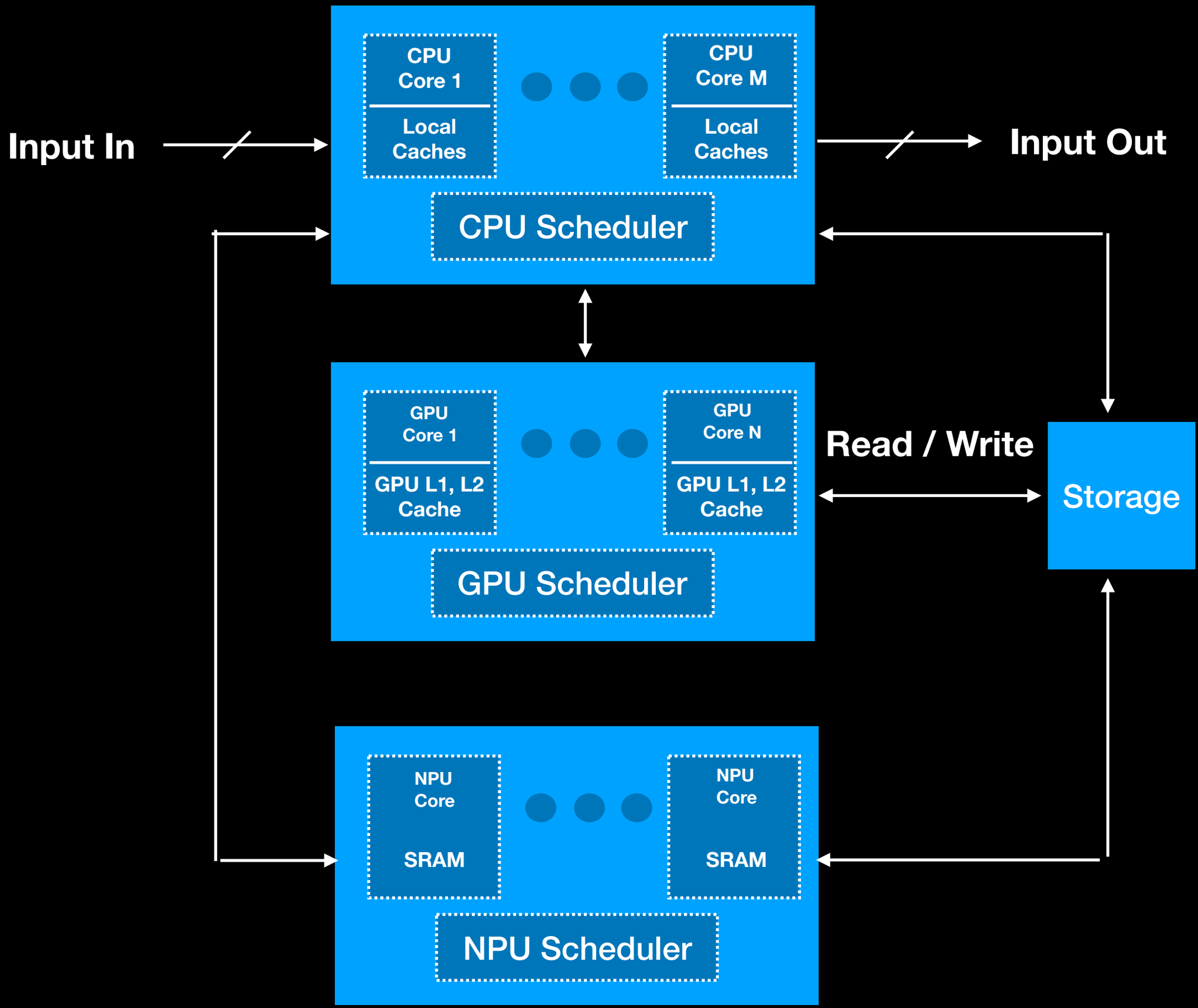


High Level Evolution



“The compute engine and the task specific engines are BOTH inefficient / ill suited, for my new application. We really need to do it like this = NEW ….”


Input In
Compute Engine 1
Local Storage 1
Compute Engine M
M Local Storage N
Input Out
Scheduler
TS Compute Engine 1
TS Local Storage 1
TS Compute Engine N
TS Local Storage N
Read / Write
Storage
TS Scheduler
New Compute Engine 1
New Local Storage 1
New Compute Engine O
New Local Storage O
New Scheduler



In 2025 / 2026, it now looks like this. On mobile devices, we are memory bound = “We have adequate compute. We need to make sure that operators (compute) have operands (data) to process.”

Apple CoreML

- CoreML acts a director, converting the learned model, and determining what portions of the computational graph will be computed “where”.

- Computation can be performed exclusively in the Apple Neural Engine (ANE), or, across the CPU, GPU and NPU.

- Ideally, the developer wants his / her model to “fit” perfectly, resulting in ONLY ANE utilization.

- Necessary conditions a perfect “fit” include:

Apple CoreML

- i.e. Does the model only use supported ANE operators?
- i.e. Is the model static (vs dynamic)? / Are the tensor sizes static and fixed at compile time?
- i.e. Is the scratch ram (SRAM) adequate for the deployed model (vs DRAM fetches)?
- i.e. Does the model perform well using only INT8 and INT16 (vs FP16 or FP32)?

Ideal Accelerator Data Flow

- A pipeline of supported ML ops
- With only SRAM memory access and repeated data reuse
- With static tensor shapes
- And low precision

Example

- MAC
-

Details-Inner workings

CNN vs Transformer
On NPU

- Why are transformers not well suited for the current NPU accelerators in 2025?

CNN vs Transformer
On NPU

- One of the big advantages of a Transformer vs a CNN is the attention mechanism.
- CNN’s have “local” attention, from its receptive field.
- Transformers have global attention, since every patch or token attends (Query / Key) to every other patch or token.
- However, global attention is expensive to realize on existing NPU’s.

CNN vs Transformer
On NPU

- Workaround: “local transformer attention”

- However, this then undermines the inherent power of a transformer.

CNN vs Transformer On NPU

- Another advantage of transformers over CNN's is the improved model performance on vision tasks, for very large transformer models.

- However, these large models require a significant amount of memory.

- Unfortunately, SRAM memory is very limited on existing NPU's.

- If DRAM fetches are required, performance degrades.

CNN vs Transformer
On NPU

- Workaround: Smaller transformer models
- Workaround: Reduced precision
- However, model performance degrades, eliminating the “upside” of using a transformer.

Related Transformer Architecture Challenges

- Matrix multiplication, whose shape changes, depending on varying / dynamic input token lengths.
- Persistent mutable state.
- Growing memory footprint.
- Expensive softmax at every layer in the transformer vs only at the final layer in a CNN classifier.

CNN vs Transformer

- CNN’s map cleanly to existing NPU’s.
- Transformers do not.
- Statefulness of a transformer vs statelessness of a CNN.
- Bandwidth heaviness of a transformer vs light bandwidth lightness of a CNN.