Simulating Time With Square-Root Space

image courtesy of DALL-E

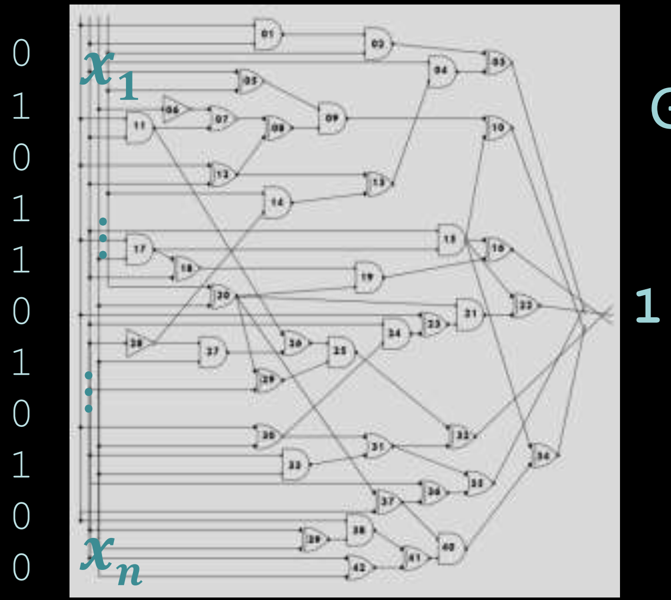
Ryan Williams, MIT and IAS (Princeton)

How space-efficiently can one simulate time-efficient computations?

Every $T(n)$-time program will use no more than (about) $T(n)$ space.

Given any $T(n)$-time program M,
is there always a way to reimplement M, so it uses $\ll T(n)$ space?

Canonical Problem: **Circuit Evaluation / Circuit Value Problem (CVP)**



Given C, x compute the output

On circuits of size s:

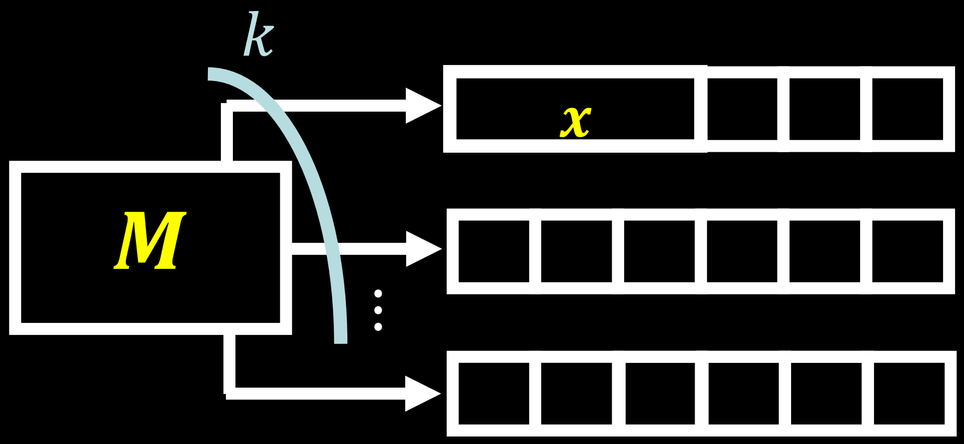
- about s time to solve CVP
- needs $\Omega(s)$ space, to store intermediate gate values?

[PV'76, Borodin'77]
CVP is in $O(s/\log s)$ space

[HPV'75, PR'81, HLMW'86]
TIME[t] $\subseteq$ SPACE$[t/\log t]$

"Pebbling approach" with an $\Omega(t/\log t)$ lower bound [LT'79] *(more later on this)*

Model of Computation: Multitape Turing Machine



- read/write k cells in each step
- the k tape heads can move left/right, in each step

TIME$[T]$:= problems solvable in $O(T(n))$ time on a MTM [HS'65]

Old model, but ***very*** *robust!*

Examples: CVP, Sorting, FFT are in TIME$[n \cdot polylog(n)]$, $n^{3-\varepsilon}$-time matrix mult algorithms are in TIME$[n^{3-\varepsilon} \cdot polylog(n)]$, etc.

(in fact, two tapes suffice [HS'66])

Open Problem: find ***any*** problem solvable in $O(n)$-time on RAMs with no $O(n \cdot polylog(n))$-time MTM

Theorem: For all $T: \mathbb{N} \to \mathbb{N}$ with $T(n) \geq n$,

$$\mathsf{TIME}[T] \subseteq \mathsf{SPACE}[\sqrt{T \log T}]$$

Every MTM running in time T has an equivalent MTM using $O(\sqrt{T \log T})$ space (!!)

(Now False) Conjecture: For all $\varepsilon > 0$, TIME[T] $\not\subseteq$ SPACE[$T^{1-\varepsilon}$]

[Sipser'86] Conjecture + Explicit Expanders ⇒ P = RP

(nowadays, we know [NW'94,IW'97,…] that we "only" need *circuit lower bounds* on time in order to achieve derandomization; we don't need *space lower bounds* on time)

Corollary: CVP ∈ SPACE[$\sqrt{n} \cdot polylog(n)$]

By diagonalization, we can conclude new lower bounds (towards $P \neq PSPACE$):

Corollary: For all "nice" $s(n) \geq n$, SPACE[s] $\not\subseteq$ TIME[$s^2 / \log^2(s)$]

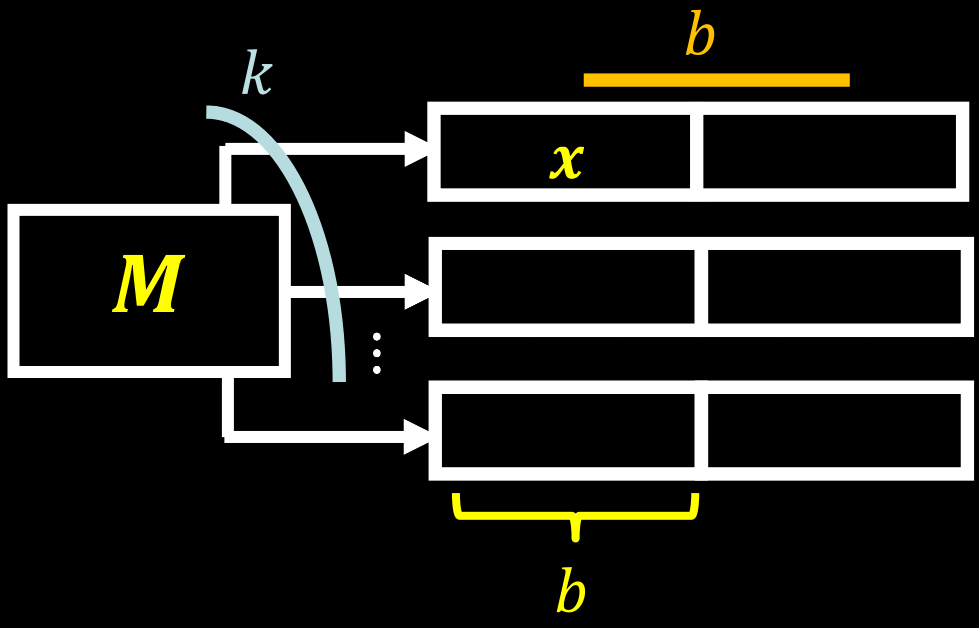
Corollary: There is an explicit Π ∈ SPACE[n] not in TIME[$n^2 / \log^2 n$]

Can the Theorem be improved?

Proposition: If TIME[T] ⊆ SPACE[T^{ε}] for all $\varepsilon > 0$, then $P \neq PSPACE$

Prior Work: TIME$[T]$ ⊆ SPACE$[T/\log T]$

Given MTM M, input x of length n
Partition the k tapes of M into **blocks** of b cells



For simplicity: $T(n) \geq n^2$, $b \approx \sqrt{T(n)}$

Define a computation graph $G_{M,x}$:

nodes: $\{0,1,\ldots,T(n)/b\}$

each node represents a **time interval**: b steps of time

edges: for $i < j$, edge (i,j) if *"info computed in interval i is needed to compute info in interval j"*

$\forall i$, $(i-1,i)$ is an edge

need the last state from interval $i-1$, to start interval i

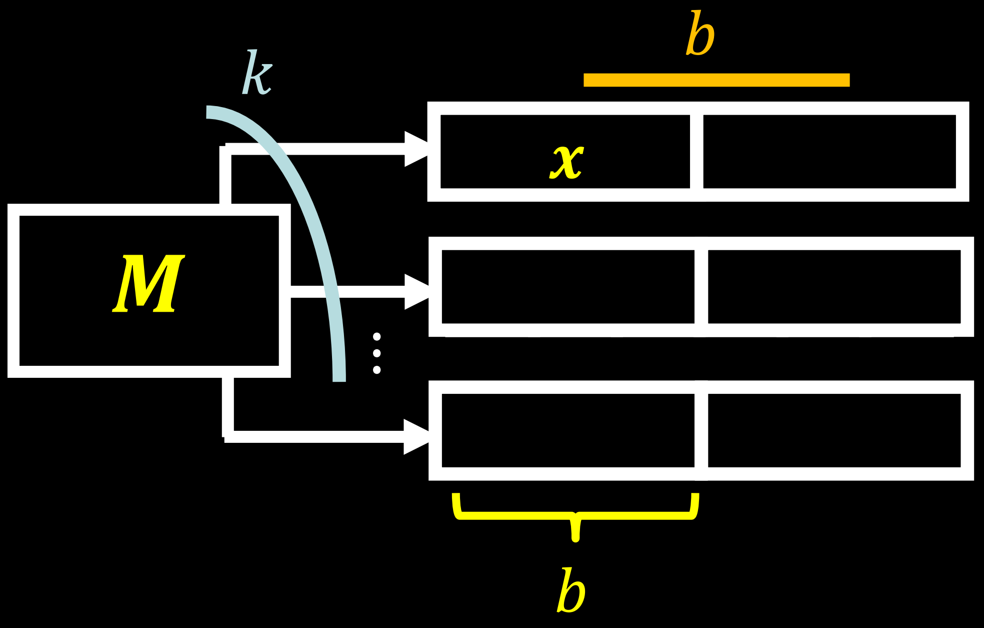
for $i < j$, edge (i,j) if some block is visited in intervals i and j but is not visited in intervals $i+1,\ldots,j-1$

Obs: during an interval, each tape has ≤ 2 blocks accessed

$$\forall i, \ \text{indeg}(i) \leq \underbrace{2k}_{\text{from blocks}} + \underbrace{1}_{\text{from state}}$$

Prior Work: TIME$[T] \subseteq$ SPACE$[T/\log T]$

Given MTM M, input x of length n
Partition the k tapes of M into **blocks** of b cells



For simplicity: $T(n) \geq n^2$, $b \approx \sqrt{T(n)}$

computation graph $G_{M,x} = (V, E)$

$V = \{0,1,\ldots,T(n)/b\}$

for $i < j$, $(i,j) \in E$ iff *"info computed in interval i is needed to compute info in interval j"*

$\forall i$, indeg$(i) \leq 2k+1$

Define strings with the info computed in each interval:

content(0): initial configuration of M on x $[O(n)$ bits$]$

content(i): info computed during interval i $[O(b)$ bits$]$

[state + head positions at end, tape blocks accessed]

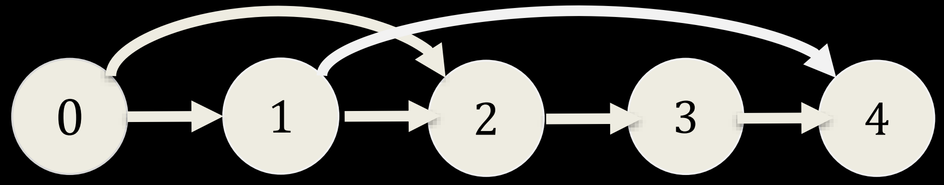
Obs: given content(i) for all i with $(i,j) \in E$, can compute content(j) in $O(b)$ time

Goal: Determine content$(T(n)/b)$

[contains accept/reject state!]

Prior Work: TIME[T] ⊆ SPACE[$T/\log T$]

Now we have a problem on DAGs:



Pebble Game: played on v-node graph

1. Can put pebble on source
2. Given pebbles on all i with $(i,j) \in E$, can put pebble on j
3. Can remove pebbles at any time

How many pebbles needed to put a pebble on the last node? [HPV'75,LT'79]

$\Theta(v/\log v)$ for DAGs of $O(1)$ indegree

Store only $O(T/b)/\log(T/b))$ content strings at a time ⇒ $O(T/\log(T/b))$ space

computation graph $G_{M,x} = (V, E)$

$V = \{0,1,\ldots,T(n)/b\}$

for $i < j$, $(i,j) \in E$ iff "*info computed in interval i is needed to compute info in interval j*"

$\forall i$, indeg$(i) \leq 2k + 1$

Define strings with the info computed in each interval:

content(0): initial configuration of M on x [$O(n)$ bits]

content(i): info computed during interval i [$O(b)$ bits]

[state + head positions at end, tape blocks accessed]

Obs: given content(i) for all i with $(i,j) \in E$, can compute content(j) in $O(b)$ time

Goal: Determine content($T(n)/b$)

[contains accept/reject state!]

Towards Low Space: The Tree Evaluation Problem (TEP)

[Braverman-Cook-McKenzie-Santhanam-Wehr'09]

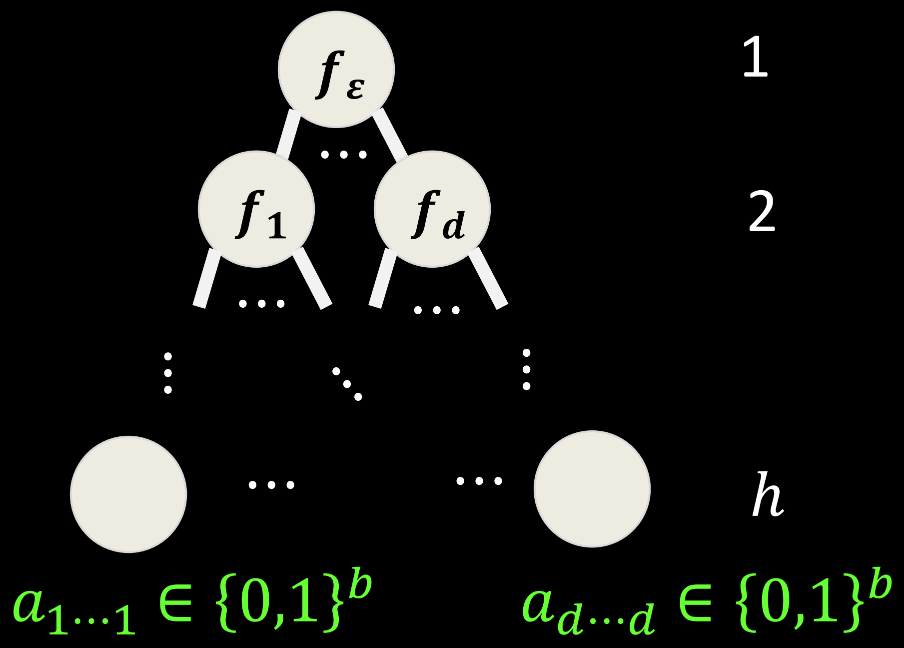
Parameters $d, b, h > 0$. Given: d-ary tree of height h

Each inner node u is labeled by $f_u: \{0,1\}^{d \cdot b} \to \{0,1\}^b$

[each f_u is given as a table of length $2^{d \cdot b}$]

Each leaf ℓ is labeled by a value $a_\ell \in \{0,1\}^b$ [d^h values]

For an inner node u, if $a_{u\,1}, \dots, a_{u\,d}$ are the values of the children of u, then value of u is $a_u := f_u(a_{u\,1}, \dots, a_{u\,d})$



Goal: Find value of the root: $a_\varepsilon \in \{0,1\}^b$

Simple recursive algorithm:

TE(v): If v is a leaf, return a_v

For $i = 1, \dots, d$, compute $A_i = \text{TE}(v_i)$

Return $f_v(A_1, \dots, A_d)$ and erase $A_1, \dots, A_d$

Takes $O(h \cdot d \cdot b)$ space $\to O(\log^2 N)$ for input length N

How much space?

[Braverman et al] showed LBs in restricted settings, conjectured TEP $\notin$ NL

Towards Low Space: The Tree Evaluation Problem (TEP)

[Braverman-Cook-McKenzie-Santhanam-Wehr'09]

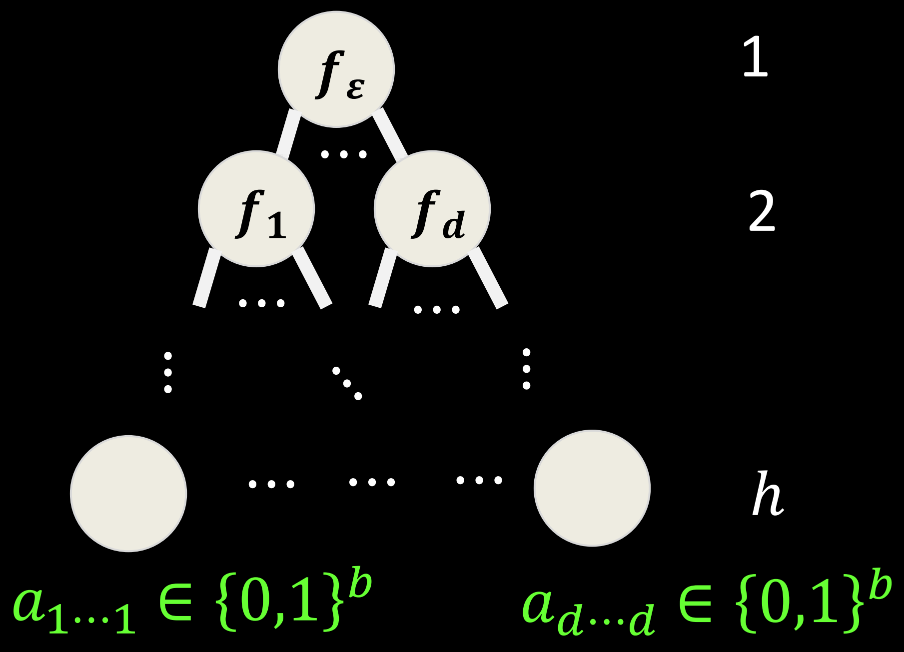
Parameters $d, b, h > 0$. Given: d-ary tree of height h

Each leaf ℓ is labeled by a value $a_\ell \in \{0,1\}^b$

Each inner node u is labeled by $f_u: \{0,1\}^{d \cdot b} \to \{0,1\}^b$

Goal: Find value of the root: $a_\varepsilon \in \{0,1\}^b$

Simple algorithm takes $O(hdb)$ space → stack of height h



[Cook-Mertz'24] TEP in $O(h \log(db) + db)$ space!

Stack of height h, with only $O(\log(db))$ bits on each level

Key Idea: The simple algorithm allocates fresh space for each value computed at each level

Instead, XOR values into existing space! Algorithm XORs the value of u into existing space, assuming an oracle that can XOR the values of u's children into existing space.

Theorem: For $T: \mathbb{N} \to \mathbb{N}$ with $T(n) \geq n^2$, TIME$[T] \subseteq$ SPACE$[\sqrt{T} \cdot \log(T)]$

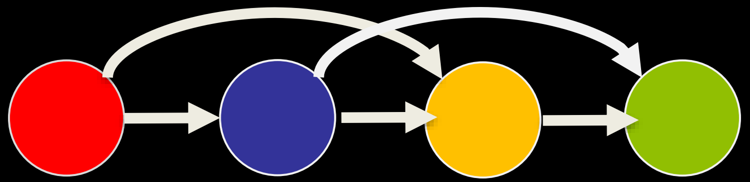
Idea: Given time-T MTM M and input x, reduce to (exp-sized) TEP instance

Assume we know $G_{M,x}$ [either it's stored in memory, or we can compute its edges efficiently]

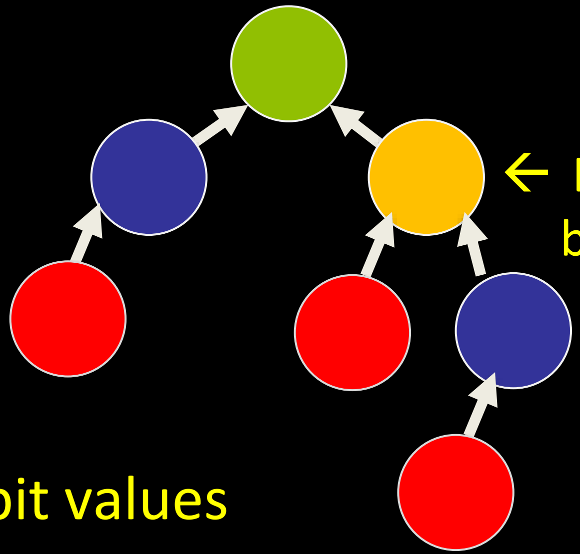
Set $d := 2k + 1$, $h := 1 + T(n)/b$, and set b so that for all i, |content(i)| $\leq b$

Extend Cook-Mertz for trees with $\leq d$ children at each node, and max height h

[Folklore] Boolean circuits of depth h $\quad\mapsto\quad$ Boolean formulas of depth h



$\mapsto$



← Each gate is indexed by a path from a gate in the circuit to the output gate

This transformation works just as well for circuits and formulas with b-bit values on the wires

Computing content in $G_{M,x}$ is evaluating a circuit over b-bit values

TEP $\equiv$ Evaluating a formula over b-bit values of depth h and fan-in d

Theorem: For $T: \mathbb{N} \to \mathbb{N}$ with $T(n) \geq n^2$, TIME$[T] \subseteq$ SPACE$[\sqrt{T} \cdot \log(T)]$

Idea: Given time-T MTM M and input x, reduce to (exp-sized) TEP instance

Assume we know $G_{M,x}$ [either it's stored in memory, or we can compute its edges efficiently]

Set $d := 2k+1$, $h := 1 + T(n)/b$, and set b so that for all i, |content(i)| $\leq b$

We can evaluate content$(T(n)/b)$ of $G_{M,x}$ using the Cook-Mertz procedure!

Space of Cook-Mertz: $O(d \cdot b + h \cdot \log(d \cdot b)) \leq O(b + (T \log b)/b)$

set b to minimize $\to O(\sqrt{T \log T})$ space

BUT! We don't know $G_{M,x}$ …

Easiest fix: Program M so that we can calculate all head positions quickly (in advance)

Def. M is oblivious if $\forall n$ and x of length n, head movements of M on x only depend on n

Thm [HS'66,PF'79] For all MTM M in $T(n)$ time, there's an ***equivalent*** two-tape oblivious M' that runs in $O(T(n) \log T(n))$ time. Given n, i in binary, head positions of M' at step i can be computed in $\boldsymbol{poly(\log T(n))}$ **time**. → Can calculate $G_{M',x}$! (But we lose a little)

Open Problems

- Extend to RAMs? (Show $TIME_{RAM}(T) \subseteq SPACE(T^{1-\varepsilon})$?)
- Extend to Parallel Machines? (Show $TIME(T) \subseteq ATIME(T^{1-\varepsilon})$?)
  Would imply super-linear time lower bounds for problems like Quantified Boolean Formulas!
- $TIME(T) \subseteq SPACE(T^{0.5-\varepsilon})$? *This would resolve the first two!*

  Improve this bound for single-tape Turing machines!
- *Barrier?*

  I'm very lucky that I found this reduction *after* Cook-Mertz, or I would have definitely declared the reduction to be a "barrier"…

"You can't solve Tree Eval in much less space, or you'd improve HPV…

…which we **know** *you can't do"*

Thank you!

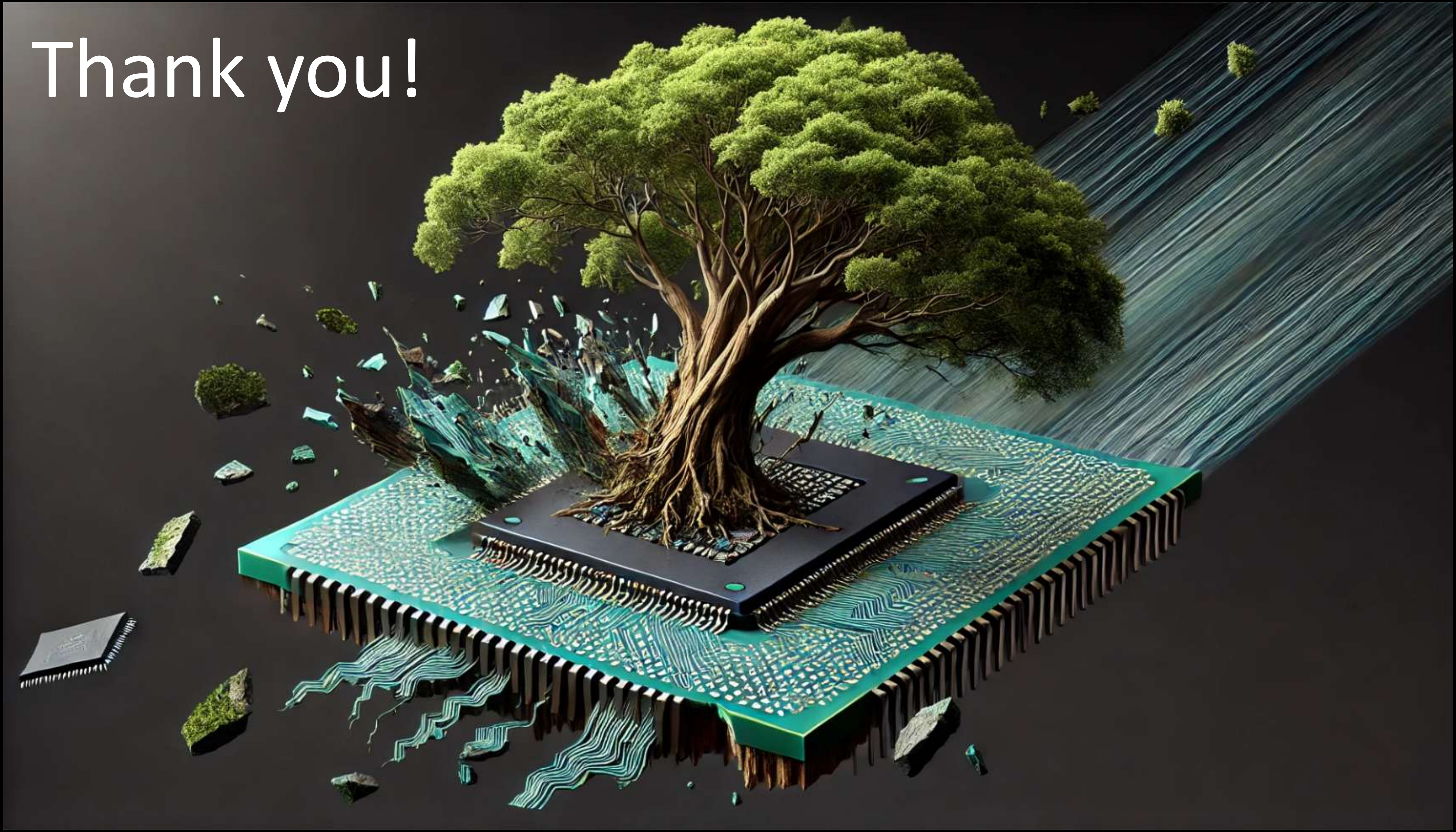


image courtesy of ChatGPT