拓撲約束執行引擎 v2.0

2026 年 7 月 30 日 / 2026-07-30 / Neo.K（許筌崴） / 公開概念技術論文／系列統合總綱／可驗證編譯—執行架構提案

https://thisoneisneok.com/html/papers/ccp-03.html

SECTION

從「無限維計算流形」到圖重寫、需求導向躍遷與異質資源映射

英文題名： Topology-Constrained Execution Engine v2.0: Graph Rewriting, Demand-Directed Leap Execution, and Heterogeneous Resource Mapping

文件編號： EML-TCE-2026-v2.0

作者： Neo.K（許筌崴）

協作修訂： Aletheia

機構： 一言諾科技有限公司（EveMissLab），台灣

原始版本： 2026 年 4 月 4 日

公開修訂版： 2026 年 7 月 30 日

文件性質： 公開概念技術論文／系列統合總綱／可驗證編譯—執行架構提案

建議縮寫： TCE（Topology-Constrained Execution Engine，拓撲約束執行引擎）

證據等級： E0——架構與形式模型；尚未完成 TCE 軟體原型、硬體後端或第三方重現

系列定位： O-Chip、SynCore、DEO、DPCPS、DryCore、SDMCA、AetherGlass–LaserCPU 與 RMC/LRTC 之上的計算表示、重寫、映射與驗證層

SECTION

修訂聲明

本文由《拓撲計算引擎：無限維幾何約束下的計算本體論革命》重構而來。原稿最有價值的洞察，是拒絕把計算只理解為「依固定順序在固定處理器上執行一串指令」，並提出三個值得保留的問題：

- 同一項語義工作是否可以具有多種等價的計算結構？
- 計算圖、資料佈局與硬體互連拓撲是否應共同最佳化？
- 面對局部查詢時，系統是否可以只啟動與結果相關的因果閉包，而不執行整個全域圖？

這三個問題分別對應程式重寫空間、拓撲感知映射、需求導向局部執行，確實構成一個可以實作的編譯—執行研究方向。

然而，原稿同時將數學隱喻、物理量子論、編譯器、運行時與硬體互連混合成單一「終極架構」，並提出多項未建立的定理與性能結論：

- 將序列、並行、拓撲與「概念空間」指定為固定的 1、3、5、無限維階梯；
- 宣稱可操作維度越多，計算能力必然越高；
- 將點、線、弦、歪線、波、場、面、體直接對應到 O(n)、O(log n) 或 O(1)；
- 將所有計算表示成么正量子演化，並引入沒有物理或形式語義的「計算普朗克常數」；
- 宣稱「插隊計算」可以把一般 DAG 的計算由 O(n) 降至 O(Query) 或 O(log D)；
- 將二分取樣誤認為完成所有離散點計算，或將解集的符號表示誤認為已枚舉所有元素；
- 將 AI 描述成可直接選出唯一最優計算流形的「幾何雕塑家」；
- 將 CXL 延伸成無限維、無限頻寬、可單週期融合任意 CPU、GPU 與記憶體的 CXL_∞；
- 將跨裝置連通誤寫成共享暫存器、算力與記憶體可以無代價相加；
- 宣稱同一任務可獲得 10^3 至 10□ 倍普遍加速，並將這些推演視為「不同維度」而非需要基準驗證的性能主張。

v2.0 因此進行以下根本重構：

- 將 TCE 從 Topological Computing Engine 改為 Topology-Constrained Execution Engine。「拓撲」指程式、資料、資源、互連與故障域之間的有限關係結構，不指超越圖靈計算的神秘無限維物理。
- 將無限維改為有限可表示的高維約束空間。 任務可具有很多語義、形狀、佈局、精度、功率與可靠度特徵，但任何實際編譯與控制都使用有限資料結構、有限精度與有限搜尋預算。
- 將幾何約束譜改為計算形態庫。 點、路徑、DAG、圖、超圖、網格、張量、胞腔複形、場與狀態空間是可選表示；它們不自動決定複雜度，也不存在普遍的維度—性能單調律。
- 將「AI 拓撲塑形」改為受約束的等價重寫與成本模型。 系統可用規則、e-graph、搜尋、整數規劃、啟發式或學習模型提出候選方案，但每個方案必須通過語義合法性、記憶體一致性、資源與安全驗證。
- 將「插隊計算」正式改寫為需求導向躍遷執行。 系統只執行查詢輸出的祖先閉包、稀疏前沿或事件觸發區域；它可以減少實際工作量，但不能跨越真實資料依賴、不可忽略的副作用或資訊理論下界。
- 將 CXL_∞ 改為有限資源織網。 CXL 提供系統級記憶體與裝置互連、池化及一致性能力；其頻寬、延遲、拓撲、交換、RAS 與擁塞均有限，不能取代片上旁路、暫存器網路或低延遲核心融合互連。
- 將 SynCore 降為執行後端之一。 TCE 可將不同子圖映射到 CPU、GPU、SynCore Fusion Cluster、Digital Flow Array、光子線性單元、物理儲備池或遠端資源，但不假定所有後端可以任意融合。
- 正式區分編譯期、部署期與運行期。 等價重寫、資料格式選擇和大規模佈局通常在編譯或部署階段完成；運行期只在決策時限允許時執行局部映射、躍遷、遷移與回退。
- 加入證明義務、確定性重播、版本化索引、模型不確定性與安全回退。 AI 只能提出計畫，不能免除正確性檢查。
- 刪除所有固定加速倍率與「後圖靈」宣稱。 TCE 在一般情況下仍受相同可計算性、依賴鏈、通訊與資訊下界約束；其價值必須來自避免無效工作、探索更好等價表示、降低資料移動與提高異質硬體利用率。

因此，新版保留的核心不再是「計算即存在，幾何即命運」，而是更精確且可證偽的命題：

> 程式語義、資料結構、依賴關係與硬體拓撲可以被共同表示與重寫；若系統能在有限成本內找出更適合當前查詢與資源狀態的等價拓撲，便可能減少實際執行工作、資料搬移與控制衝突。

SECTION

摘要

現代計算系統已不再是單一 CPU 上的線性指令序列。機器學習模型、資料庫查詢、圖分析、科學計算、編譯工作流與分散式服務，通常以有向無環圖、控制流圖、資料流圖、稀疏張量、超圖、狀態機或多階段管線表示；執行資源也同時包含 CPU、GPU、NPU、DPU、chiplet、近端與遠端記憶體、CXL 資源池、可重構資料流陣列、光子或物理計算後端。真正的最佳化問題不只是「把某個 kernel 跑得更快」，而是決定：哪一種等價程式表示、哪一種資料格式、哪一個依賴切分、哪一個資源映射、哪一條資料移動路徑，以及哪一個運作包絡，能在正確性、延遲、能耗、容量、可靠度與重組成本之間取得可驗證的平衡。

本文提出拓撲約束執行引擎 TCE。TCE 將工作負載表示為具有資料、控制、副作用、形狀、精度與品質約束的多關係計算圖，將硬體表示為具有頻寬、延遲、容量、功率、溫度、健康度與故障域的資源圖，再建立候選等價重寫空間。系統可使用規則式圖重寫、e-graph 等式飽和、稀疏張量格式變換、資料流切分、算子融合、重計算、快取與遠端資源選擇，產生多個合法執行拓撲；之後由成本模型與驗證器共同選擇計畫。

本文重新定義原稿的「插隊計算」為需求導向躍遷執行（Demand-Directed Leap Execution, DDLE）。對一組查詢輸出 Q，系統先計算其依賴祖先閉包 A(Q)，再只啟動該閉包內尚未物化且確實需要的節點。若計算是純函數式、依賴圖完整、外部副作用已被顯式建模，則執行 A(Q) 足以產生與全圖執行相同的 Q；其工作量上界由查詢相關子圖大小決定，而不是由全圖大小決定。但索引建置、快取維護、依賴分析與失效重算仍有成本，且在 A(Q) 接近全圖、存在全局副作用或依賴鏈很長時，DDLE 不提供漸近優勢。

TCE 使用 TopoIR 作為多層中介表示，包含語義圖、等價重寫圖、部署圖、運行時前沿與證據圖。其七層架構依序為：語義與效果層、等價重寫層、依賴與局部性層、拓撲映射層、需求導向運行時層、異質執行與精確提交層，以及證據與治理層。O-Chip 負責部署後的意圖與跨層編排；SynCore、CPU、GPU、AetherGlass–LaserCPU 或 RMC/LRTC 提供受限的執行域；DEO、DPCPS、DryCore 與 SDMCA 分別提供運作、功率、熱與空間資源證書。

本文不宣稱 TCE 可以把任意 O(n) 問題變成 O(1)，不宣稱更高表示維度必然帶來更高計算能力，也不宣稱 CXL 可建立無限頻寬拓撲。TCE 的可檢驗主張是：對具有多種等價表示、稀疏依賴、局部查詢、昂貴資料移動與異質後端的工作負載，圖重寫、查詢閉包、拓撲感知映射與證據化回退可能降低實際執行節點數、資料搬移、尾端延遲或牆插能耗。若在嚴格基線、完整成本與第三方重現下不能獲得優勢，則相應機制應被否證或縮減。

關鍵詞： 圖重寫、e-graph、等式飽和、多層中介表示、需求導向執行、稀疏計算、拓撲感知映射、異質運算、CXL、資源圖、資料流、可驗證編譯、O-Chip、SynCore

SECTION

1.1 從指令序列到多關係計算結構

一段程式可以同時具有多種結構：

- 語法樹描述語言構造；
- 控制流圖描述可能的執行路徑；
- 資料依賴圖描述 producer–consumer 關係；
- 記憶體圖描述別名、生命週期與一致性；
- 任務圖描述粗粒度階段；
- 張量與稀疏格式描述資料形態；
- 部署圖描述算子、資料與裝置位置；
- 遙測圖描述功率、熱、健康與故障關係。

因此，比「計算究竟是線、面或場」更有用的問題是：

> 在哪一層、對哪一種語義與成本，我們應該使用哪一種有限表示？

TCE 不把這些表示視為同一物件的神秘維度，而視為互相投影、保留不同不變量的多層 IR。

SECTION

1.2 三種不應混淆的維度

新版區分三種維度。

物理維度

元件與互連實際存在於一維、二維或三維空間，並受有限傳播速度、寄生、熱與製造約束。

表示維度

一個節點可以具有 d 個有限特徵：

> $$z_v = (z_{v1}, z_{v2}, \ldots, z_{vd}).$$

d 可以很大，但仍是模型設計與資料結構選擇，不代表硬體進入額外物理維度。

組合維度

當系統允許多種切分、佈局、格式、排程與後端時，候選計畫數可能呈組合爆炸：

> |Ω|
> =
> |Ω_(rewrite)|
> |Ω_(layout)|
> |Ω_(mapping)|
> |Ω_(schedule)|.

原稿所感受到的「高維」更接近這種大規模離散搜尋空間，而不是可直接提供 O(1) 解答的無限維流形。

SECTION

1.3 維度不等於能力

對同一項任務，增加特徵、連接或自由度可能擴大可搜尋方案，也可能增加：

- 訓練與搜尋成本；
- 校準負擔；
- 噪聲和串擾；
- 過擬合；
- 可驗證性困難；
- 執行時控制成本。

  因此不存在普遍關係：

> 計算能力
> ∝
> 表示維度.

更合理的是：

> U
> =
> B_(task)
> -
> C_(search)
> -
> C_(execute)
> -
> C_(verify)
> -
> R_(failure),

其中 B_(task) 是表示自由度對特定任務帶來的收益。

SECTION

1.4 TCE 不改變可計算性類別

TCE 是編譯、部署和運行時架構，不宣稱超越圖靈可計算性。它可以：

- 用符號形式取代不必要的枚舉；
- 只執行查詢相關子圖；
- 利用稀疏性與等價重寫；
- 將工作映射到適合的異質後端；
- 以近似方法換取可量化誤差；
- 重用已物化結果。

  但它不能：

- 無代價解決不可判定問題；
- 省略輸出本身要求的資訊量；

- 跨越真實因果依賴；
- 將所有組合搜尋壓成常數時間；
- 用表示上的「高維」取消物理通訊與記憶體成本。

SECTION

2.1 工作負載圖

TCE 將工作負載表示為多關係有向超圖：

> cal-G_W
> =
> (
> V_W,
> E_D,
> E_C,
> E_M,
> E_S,
> a,
> q
>).

其中：

- V_W：算子、任務、狀態轉移或外部動作；
- E_D：資料依賴；
- E_C：控制依賴；
- E_M：記憶體、別名與一致性關係；
- E_S：副作用與順序關係；
- a：形狀、型別、稀疏度、精度、估計成本等屬性；

- q：期限、品質、隱私、安全與可重現要求。

  只用普通 DAG 不足以表達循環、可變狀態與副作用，因此 TopoIR 允許區域、迭代、狀態端口和效果 token。

SECTION

2.2 資源圖

硬體與系統資源表示為：

> cal-G_R(t)
> =
> (
> V_R,
> E_R,
> m_R(t),
> f_R(t)
>).

其中：

- V_R：CPU 核心、GPU、NPU、SynCore 執行域、記憶體、CXL 裝置、網路、儲存與控制器；
- E_R：互連及可達關係；
- m_R(t)：頻寬、延遲、容量、功率、溫度、健康度與佔用；
- f_R(t)：故障域、隔離邊界與信任等級。

  TCE 只映射到當下被證書允許的資源集合：

> V_R^(safe)(t)
> ⊆
> V_R.

SECTION

2.3 等價重寫圖

對一個語義區域 p，系統建立候選表示集合：

> $$[p]_{\equiv} = \{p' \mid p' \equiv p\}.$$

這些等價式可以由：

- 代數律；
- 算子融合與分解；
- 迴圈交換、分塊和向量化；
- 稀疏格式等價；
- 張量重排；
- 重計算與物化；
- 查詢下推；
- 圖演算法的 semiring 表示；
- 受控近似規則；

  產生。

  TCE 可用 e-graph 壓縮表示大量等價式，而不是每次立即承諾一條局部最佳化路徑。

SECTION

2.4 執行計畫

一個候選計畫定義為：

$$
\Pi
=
(\rho,\mu,\lambda,\sigma,\delta,\phi),
$$

其中：

- ρ：選定的重寫與分解；
- μ:V_W→ V_R：任務—資源映射；
- λ：資料格式、佈局與位置；
- σ：排程、同步與前沿策略；
- δ：DEO 運作包絡與功率—熱請求；
- φ：失敗、回退與結果驗證計畫。

成本函數不是只有時間：

$$
\begin{aligned}
&J(\Pi)\\
&=\\
&\alpha\, L_{(end)}\\
&+\beta\, L_{(p99)}\\
&+\gamma\, E_{(wall)}\\
&+\delta\, D_{(move)}\\
&;\\
&\ +\eta\, C_{(reconf)}\\
&+\zeta\, R_{(failure)}\\
&+\xi\, U_{(model)}\\
&+\rho\, C_{(verify)}.
\end{aligned}
$$

SECTION

2.5 證據圖

每次計畫必須連接到證據：

> cal-G_E
> =
> (V_E,E_E),

其節點包含：

- 重寫規則版本；
- 型別、形狀與效果證明；
- 成本模型版本；
- 校準資料；
- 基準結果；
- 硬體與韌體版本；
- 執行日誌；
- 失敗與回退紀錄。

沒有證據血統的 AI 計畫不能被當成可重現編譯成果。

SECTION

3.1 形態不是複雜度類別

新版保留「點、線、波、場」作為設計語彙，但取消它們與複雜度的一對一對應。更合理的形態庫如下。

形態	正式表示	適合問題	常見限制
點	純量算子／狀態	局部轉換、標量控制	並行度有限
路徑	pipeline／sequence	串流、協議、階段式工作	受最慢階段限制
DAG	dataflow／task graph	模型推論、編譯、科學工作流	關鍵路徑與排程開銷
圖	adjacency／incidence	網路、稀疏關係、遍歷	不規則訪存與負載不均
超圖	多輸入多輸出關係	編譯、資料庫、張量 contraction	分割與映射困難

網格／張量	index space	密集線性代數、stencil、影像	資料搬移與維度爆炸
胞腔／單純複形	cells／simplices	高階關係、拓撲特徵	工具與後端有限
場／算子	PDE／operator	物理模擬、神經算子、光學	離散化與穩定性
狀態空間	transition system	控制、協議、動力系統	狀態爆炸
語義空間	typed symbolic IR	推理、規則與程式變換	等價判定與搜索成本

SECTION

3.2 形態轉換

TCE 關心的不是「哪個形態比較高維」，而是轉換是否保留所需語義。

例如，圖演算法可以映射到稀疏矩陣與 semiring 運算：

> cal-G_(graph)
> xrightarrowcal-T_(GB)
> (A,⊕,⊗).

張量表達式可以映射到不同儲存格式和迭代空間：

> cal-E_(tensor)
> xrightarrowcal-T_(format)
> (λ_(CSR),λ_(COO),λ_(CSF),⋯).

控制流區域也可以在特定條件下轉成資料流或 predication；但若轉換改變例外、I/O、副作用或浮點誤差，就必須在語義合約中顯式說明。

SECTION

3.3 形態選擇不是唯一解

原稿假設每個任務存在唯一最優流形。實際上，成本模型通常非凸、硬體狀態會改變、量測有噪聲，且多個計畫可能位於 Pareto 前沿。

定義：

> Π⊠precΠ⊠

表示 Π⊠ 在所有關鍵指標不差於 Π⊠，且至少一項更好。TCE 尋找的是：

> cal-P^(*)
> =
> {Πmid not exists Π'precΠ},

而不是假裝存在可由 AI 一次找出的宇宙唯一解。

SECTION

3.4 形態與後端的多對多關係

同一 DAG 可以在 CPU、GPU、FPGA、SynCore Digital Flow Array 或分散式 runtime 上執行；同一 GPU 也能執行張量、圖與部分控制流。形態是表示，後端是實現，兩者是多對多映射：

> cal-M:cal-F_(form)×cal-R
> →
> cal-P_(plan).

SECTION

4.1 從單一路徑最佳化到等式飽和

傳統編譯器常依固定 pass 順序執行變換；某個早期決策可能阻止後續更好的方案。等式飽和的思想是：

- 把等價表達式加入 e-graph；
- 在預算內擴展重寫空間；
- 最後依成本模型抽取候選。

形式上：

> cal-E_(k+1)
> =
> Saturate

直到達到固定點、時間、記憶體或節點數預算。

SECTION

4.2 重寫規則的四級信任

等級	規則	例子	驗證要求
R0	完全等價	整數代數、純算子融合	型別與等價證明
R1	條件等價	無別名、無溢位、形狀條件	守衛條件
R2	數值近似	低精度、重排浮點 reduction	誤差預算
R3	任務近似	剪枝、早退、代理模型	品質與風險證書

AI 生成的重寫預設為未信任候選，必須被降解到上述某一級並完成驗證。

SECTION

4.3 搜尋與抽取

候選抽取可以使用：

- 動態規劃；
- 整數線性規劃；
- beam search；
- Monte Carlo tree search；
- 貝葉斯最佳化；
- 學習成本模型；
- 人工規則和 profile-guided 搜尋。

但搜尋本身有成本：

> C_(opt)
> =
> C_(analysis)

只有當：

$$
C_{opt} < N_{reuse} \cdot \Delta C_{run},
$$

深度搜尋才具有整體收益。

SECTION

4.4 高維語義的有限表示

「概念空間」在 TCE 中不再是可直接訪問的無限維世界，而是由有限符號、型別、效果、形狀與嵌入共同描述：

$$
s(p) = (s_{symbolic}, s_{type}, s_{effect}, s_{shape}, s_{learned}).
$$

學習嵌入只能幫助檢索或排序，不構成語義等價證明。

SECTION

5.1 問題定義

設計算圖：

$$
\text{cal-}G=(V,E)
$$

以及查詢輸出集合：

> $Q \subseteq V$.

定義 Q 的祖先閉包：

> $$A(Q) = \{v \in V \mid v \text{ leadsto } q, q \in Q\} \cup Q.$$

需求導向躍遷執行 DDLE 只執行 A(Q) 中尚未有效物化的節點。

SECTION

5.2 查詢閉包正確性命題

命題 5.1（純計算閉包充分性）

若：

- cal-G 完整表示所有資料與控制依賴；
- A(Q) 中的節點為確定性純函數，或其副作用已被效果邊顯式納入；
- 所有輸入版本一致；
- 被重用的物化結果未失效；

  則執行誘導子圖 cal-G[A(Q)] 產生的 Q，與執行整個 cal-G 後讀取 Q 相同。

  此命題並未跨越依賴；它只是拒絕執行對 Q 無因果貢獻的節點。

SECTION

5.3 複雜度

若祖先索引已建立，單次查詢的圖遍歷與執行開銷可表示為：

> $$T_{query} = O(|A(Q)| + |E[A(Q)]|)$$

完整成本則為：

$$
\begin{aligned}
&T_{total} \\
&= \\
&T_{index} \\
&+T_{invalidation} \\
&+T_{query} \\
&+T_{verify}.
\end{aligned}
$$

因此不能普遍寫成 O(Q)，因為一個輸出可能依賴整個圖。

SECTION

5.4 二分取樣不等於完成所有點

對函數 f 在區間中的自適應取樣，二分可以用較少樣本建立近似：

$$
\begin{aligned}
&\hat{f} \\
&\approx \\
&f,
\end{aligned}
$$

但若任務要求輸出每個離散點：

$$
\{f(0),f(1),\ldots,f(N)\},
$$

輸出本身含有 N+1 個值，不能只靠 O(log N) 次取樣完成精確枚舉。

原稿中的 X+Y=1000 例子真正的改善，是以符號關係：

$$
\begin{aligned}
&Y=1000-X, \\
&\\
&X\in\{0,1,\ldots,1000\}
\end{aligned}
$$

壓縮表示解集；若使用者要求逐項列出 1001 組解，輸出成本仍為 Ω(N)。

SECTION

5.5 四種 DDLE 模式

DDLE-Q：查詢閉包

只計算查詢祖先，適合試算表、資料管線、建置系統與互動式模型。

DDLE-F：稀疏前沿

只啟動目前活躍 frontier：

> F_(t+1)
> =
> Γ(F⊠)\ V_(visited),

適合 BFS、稀疏圖、事件模擬與訊息傳遞。

DDLE-M：物化與增量更新

重用已物化節點，只重算受變更影響的後代閉包：

> D(Δ)
> =
> {vmid Δleadsto v}.

DDLE-P：預測式預取

根據查詢機率 P(q) 提前物化候選，但需最小化：

> J_(prefetch)
> =
> bb-E[L]
> +λ E_(waste)
> +μ C_(evict).

SECTION

5.6 失效條件

DDLE 在以下情況可能沒有優勢或產生錯誤：

- 查詢祖先接近全圖；
- 全域 reduction 或全局約束不可分解；
- 隱藏副作用未建模；
- 快取失效頻繁；
- 索引建置大於節省；
- 預測預取誤判；
- 動態圖使依賴快速改變；
- 查詢需要完整枚舉或完整證明。

SECTION

6.1 映射不是只選 GPU 編號

對任務節點 u,v，若資料量為 d_(uv)，映射後的通訊成本為：

> C_(uv)
> =
> (d_(uv))/(B_(μ(u),μ(v)))
> +L_(μ(u),μ(v))
> +C_(protocol)
> +C_(queue).

總資料移動成本：

> D_(move)(μ)
> =
> Σ_((u,v)∈ E_D)C_(uv).

因此，將兩個高互動節點放到不同 CXL 節點、遠端記憶體或不同 NUMA 域，可能比在較慢但近端的裝置上執行更差。

SECTION

6.2 圖分割與超圖分割

普通圖分割只計算成對邊；對多個消費者共享資料的情況，超圖更能表示一次資料物件被多個任務存取。

TCE 可最小化 cut：

> cut(μ)
> =
> Σ_(e∈ E_H)
> w_e
> ·
> 1[e跨越多個資源域].

但還要同時滿足容量、功率、熱、可信度和期限約束。

SECTION

6.3 關鍵路徑下界

對 DAG cal-G，即使有無限處理器，完成時間仍受關鍵路徑限制：

> T_(end)
> ≥
> CP(cal-G).

若再考慮通訊：

> T_(end)
> ≥
> CP(cal-G,μ).

TCE 可以縮短通訊、找到等價低深度圖，或把部分運算替換為符號／近似形式，但不能以「拓撲躍遷」忽略真實依賴鏈。

SECTION

6.4 多尺度拓撲

尺度	主要機制	TCE 可控制內容
核心內	pipeline、cache、OoO	通常不直接控制
片上	NoC、共享快取	kernel 與局部映射
封裝內	UCIe／專用 die-to-die	chiplet 放置、資料分區
主機內	NUMA、PCIe、CXL	記憶體與裝置映射
機箱／機櫃	SDMCA、網路、儲存	粗粒度任務與故障域
分散式	fabric／cluster	工作流、複本與資料局部性

在越細的尺度，控制時限越短，TCE 越需要預先編譯或硬體共設計；在越粗的尺度，運行時可選空間較大，但資料移動成本也更高。

SECTION

6.5 重組成本

任何拓撲重組都必須支付：

> C_(reconf)
> =
> T_(quiesce)
> +T_(checkpoint)
> +T_(move)
> +T_(program)
> +T_(warm)
> +T_(validate).

只有當預期運行收益大於該成本時，才應切換。

SECTION

7.1 AI 的三個合理角色

候選生成

根據工作負載與硬體特徵提出重寫、切分、格式與映射候選。

成本預測

預測候選在特定後端上的延遲、能耗、記憶體與失敗風險：

> (hat J,Σ_J)
> =
> cal-M_(θ)(Π,cal-G_W,cal-G_R).

搜尋策略

決定下一個應實測或模擬的候選，以減少搜尋成本。

SECTION

7.2 AI 不能替代的部分

AI 不能自行保證：

- 語義等價；
- 記憶體模型合法；
- 無資料競爭；
- 無越權 DMA；
- 功率與熱安全；

- 即時期限；
- 浮點誤差符合規格；
- 近似結果可被使用者接受。

這些由規則、形式驗證、靜態分析、硬體護欄或實驗證書處理。

SECTION

7.3 不確定性與拒絕

若模型輸出不確定性過高：

> $U(\Pi) > U_{max}$,

TCE 應：

- 使用較保守規則；
- 啟動小規模試跑；
- 回退到已知後端；
- 要求人工或離線重新編譯。

SECTION

7.4 線上學習的污染風險

運行時觀測可能受到：

- 背景負載；
- 熱狀態；
- 資料分布漂移；
- 硬體老化；

- 對抗性輸入；
- 遙測錯誤；

影響。模型更新必須版本化、可回退，並保存訓練資料血統。不能因「越用越聰明」而跳過驗證。

SECTION

8.1 CXL 的合理位置

CXL 可支援 CPU—裝置、CPU—記憶體的一致性與資源池化，適合：

- 容量擴展；
- 分層記憶體；
- 共享記憶體設備；
- 加速器與主機資料交換；
- 系統級 RAS；
- 部分資源解耦。

CXL 4.0 將鏈路速率提高到 128 GT/s，並加入 bundled ports 與更強的記憶體 RAS；這代表資源織網持續演進，但不代表頻寬、延遲或拓撲變成無限。

SECTION

8.2 CXL 不能做什麼

CXL 不適合被描述為：

- 單週期共享暫存器；
- 任意 CPU／GPU 的無延遲 ALU 融合；
- 全連接 $O(n^2)$ 通道的免費實現；
- 無限維地址空間；

- 把遠端記憶體延遲消失；
- 取消一致性、交換、序列化與擁塞成本。

SECTION

8.3 TCE 的 CXL 模型

對記憶體或裝置路徑 p：

> C_(CXL)(p)
> =
> L_(link)
> +L_(switch)
> +L_(protocol)
> +L_(queue)
> +(D)/(B_(eff)).

同時考慮：

- 跳數；
- 交換拓撲；
- 共享鏈路；
- coherency traffic；
- page migration；
- failure domain；
- RAS 與重試。

SECTION

8.4 記憶體分層與資料放置

TCE 可將資料分類為：

類型　合理位置

極熱、小型、低延遲　片上／HBM／本地 DRAM

熱、容量較大　本地 DRAM／近端 CXL

溫、可分頁　CXL 擴展記憶體

冷、可重建　遠端記憶體／儲存

共享模型或唯讀資料　CXL 池化候選

對細粒度、指標追逐、頻繁同步的資料，遠端池化可能造成嚴重延遲；TCE 必須以量測而非「維度擴張」決定放置。

SECTION

L0：語義、型別與效果層

負責：

- 型別和形狀；
- 控制與資料依賴；
- 副作用；
- 精度與品質；
- 安全與隱私標籤。

輸出語義 TopoIR。

SECTION

L1：等價重寫與形態層

負責：

- e-graph；

- rewrite rules；
- 稀疏格式；
- 算子融合／分解；
- 圖、張量、資料流與狀態空間轉換。

SECTION

L2：依賴、局部性與查詢層

負責：

- ancestor／descendant closure；
- frontier；
- incremental dependency；
- materialization；
- side-effect ordering；
- DDLE 合法性。

SECTION

L3：拓撲映射與部署層

負責：

- 工作圖—資源圖匹配；
- partition；
- data placement；
- format selection；
- CXL／NUMA／chiplet 拓撲；

- 故障域與信任域。

SECTION

L4：O-Chip／TCE Runtime 層

負責：

- 意圖與期限；
- 動態前沿；
- profile／telemetry；
- DEO 包絡請求；
- 遷移與回退；
- 決策時限管理。

SECTION

L5：異質執行與提交層

後端包括：

- CPU／GPU／NPU；
- SynCore Fusion Cluster；
- SynCore Digital Flow Array；
- FPGA／CGRA；
- AetherGlass 光子線性單元；
- RMC/LRTC 物理儲備池；
- 遠端服務或儲存計算。

結果必須經精確提交、誤差檢查或資格判定。

SECTION

L6：證據、治理與長期學習層

負責：

- 重寫證據；
- 成本模型版本；
- 基準與消融；
- 失敗資料庫；
- 可重現環境；
- 模型更新與權限；
- 第三方重現。

SECTION

10.1 O-Chip 與 TCE

> TCE
> =
> 表示、重寫、映射與驗證框架,

> O-Chip
> =
> 部署後受時限約束的跨層控制面.

TCE 可離線產生候選計畫與策略；O-Chip 在運行期依遙測選擇被允許的計畫。

SECTION

10.2 SynCore 與 TCE

SynCore 是 TCE 的異質執行後端：

- F-Mode 支援可組合純量融合；
- D-Mode 支援資料流與空間映射；
- P-Mode 支援特殊振盪器計算；
- VCE 提供驗證、提交與回退。

  TCE 不把不同模式的峰值加速相乘，而比較端到端收益。

SECTION

10.3 DEO、DPCPS 與 DryCore

TCE 計畫必須先取得：

> cal-C_(run)
> =
> cal-C_(DEO)
> ∩
> cal-C_(power)
> ∩
> cal-C_(thermal).

- DEO 提供合法運作包絡；
- DPCPS 提供功率、電流與 PDN 裕度；
- DryCore 提供溫度、冷卻與失效安全邊界。

SECTION

10.4 SDMCA

SDMCA 提供模組、互連、冷卻與故障域的空間拓撲。TCE 可把高通訊子圖映射到相鄰模組，把容錯副本分散到不同故障域，但必須計入背板、線纜、光電與服務成本。

SECTION

10.5 AetherGlass–LaserCPU

TCE 只把可合法映射的線性變換、卷積、干涉、模式匹配或特定動力學送入光子域；控制流、精確記憶、驗證與頻繁更新仍留在電子域。

SECTION

10.6 RMC/LRTC

物理計算後端被視為近似、校準依賴的特殊裝置。TCE 必須附帶：

- 輸入編碼；
- 基板狀態；
- 讀出模型；
- 不確定性；
- 數位回退；
- 能量與重置成本。

SECTION

10.7 全系列控制鏈

應用意圖
→
TopoIR

SECTION

11.1 編譯正確性

每個重寫至少檢查：

- type legality；
- shape legality；
- dominance；
- memory effect；
- alias；
- synchronization；
- exception／I/O；
- numerical error；
- approximation contract。

SECTION

11.2 記憶體一致性

跨 CPU、GPU、CXL 與加速器的資料共享必須使用正式一致性與擁有權協議，不允許以「拓撲連通」推導所有節點看見相同狀態。

資料狀態可表示為：

> Owner(x,t),
>
> Version(x,t),
>
> Scope(x,t).

SECTION

11.3 決定性重播

TCE 必須記錄：

- 計畫版本；
- 資源狀態；
- 隨機種子；
- 動態輸入；
- 重寫與映射決策；
- 失敗與重試；
- 近似後端校準狀態。

必要時重播相同語義路徑，或說明為何硬體與環境差異使結果只保證容差等價。

SECTION

11.4 看門狗與回退

若發生：

- deadline miss；
- cost model 大幅誤差；
- 熱／功率證書失效；
- CXL 裝置離線；
- 近似誤差超標；
- AI 決策域外；
- 圖索引不一致；

則回退到：

- 先前已驗證計畫；
- 平台預設編譯；
- CPU／GPU 安全後端；
- 全圖保守執行。

SECTION

11.5 安全威脅

TCE 特別需要防範：

- 惡意重寫規則；
- 成本模型投毒；
- 偽造遙測；
- CXL／DMA 越權；
- 資源耗盡；
- 以預取為名讀取敏感資料；
- 近似後端悄然降低品質；
- 共享資源側通道。

所有執行計畫都應具有能力限制與最小權限。

SECTION

12.1 MVP 目標

第一個 TCE 不製造新型處理器，也不實作 CXL_∞。它是一個軟體編譯—運行時原型：

> TopoIR＋e-graph 重寫＋資源圖成本模型＋DDLE 運行時＋CPU/GPU 後端。

SECTION

12.2 建議技術棧

- MLIR：多層 IR、dialect conversion、Transform dialect；
- e-graph／egg 或相近系統：等價空間；
- LLVM／CPU backend；
- CUDA／ROCm 或 OpenCL backend；
- GraphBLAS／SuiteSparse：圖—稀疏線性代數基線；
- TACO／MLIR SparseTensor：稀疏格式與迭代生成；
- PaRSEC／Taskflow 類 DAG runtime：動態任務基線；
- hwloc／NUMA／perf／eBPF：資源圖與遙測；
- CXL latency／bandwidth emulator：有限資源池實驗。

SECTION

12.3 工作負載

至少包含：

- 稀疏矩陣—向量與矩陣—矩陣運算；
- BFS、SSSP、PageRank 或圖前沿任務；
- 機器學習推論 DAG；
- 多階段資料處理與查詢；
- 增量建置或試算表依賴；

- 編譯工作流；
- 具有副作用與動態控制流的反例；
- 查詢祖先接近全圖的退化案例。

SECTION

12.4 基線

- 原始未最佳化版本；
- LLVM／MLIR 標準 pass pipeline；
- 單一路徑手工最佳化；
- 無 e-graph 的 greedy 重寫；
- 全圖執行；
- 查詢閉包但無物化；
- DDLE＋物化；
- 拓撲無感映射；
- NUMA／CXL 拓撲感知映射；
- 規則成本模型；
- 學習成本模型。

SECTION

12.5 量測指標

$$
K
=
(
T_{compile},
T_{run},
$$

另需記錄：

- e-graph 節點數；
- 搜尋時間；
- 成本預測誤差；
- DDLE 祖先閉包比例；
- 快取命中和失效；
- 拓撲映射切邊成本；
- 執行結果誤差；
- 回退次數。

SECTION

12.6 分階段路線

V0：形式語義

- TopoIR 核心型別；
- 效果系統；
- DDLE 閉包命題；
- 重寫信任分級。

V1：MLIR 原型

- Topo dialect；
- Transform dialect 整合；
- CPU／GPU lowering；

- 靜態成本模型。

V2：e-graph 與抽取

- 規則庫；
- 等式飽和預算；
- 多目標抽取；
- 等價測試。

V3：DDLE Runtime

- 查詢閉包；
- 增量更新；
- 物化與失效；
- 反例和副作用。

V4：拓撲感知部署

- NUMA；
- 多 GPU；
- 模擬 CXL memory tier；
- 故障域與遷移。

V5：O-Chip／SynCore 整合

- O-Soft 控制；
- SynIR／TopoIR bridge；
- DEO、DPCPS、DryCore 證書接口；

- FPGA 或 CGRA backend。

V6：第三方重現

- 開放基準；
- 固定硬體描述；
- 完整 log；
- 負結果；
- 外部實驗室重現。

SECTION

H1：等價重寫收益

對具有多種合法佈局與算子分解的工作負載，e-graph＋拓撲成本抽取在相同編譯預算下，能否優於 greedy pass pipeline？

否證條件： 編譯時間與記憶體增加，但端到端指標無顯著改善。

SECTION

H2：DDLE 工作縮減

當：

> $(|A(Q)|)/(|V|) \ll 1,$

DDLE 能否在完整索引與失效成本後，降低執行節點數和互動延遲？

否證條件： 索引、物化與失效成本抵銷全部節省。

SECTION

H3：拓撲感知資料放置

對跨 NUMA、GPU 和 CXL 模擬層級的工作負載，資源圖映射能否降低：

> D_(move)
> 或
> L_(p99)?

否證條件： 排程與遷移開銷大於局部性收益。

SECTION

H4：學習成本模型

學習模型能否在明確校準域內，比人工模型更準確地排序候選？

否證條件： 跨工作負載／硬體後失準，且拒絕機制不能控制風險。

SECTION

H5：CXL 資源池邊界

容量受限、共享唯讀或可批次搬移的工作負載，是否受益於 CXL 池化；細粒度依賴與 pointer chasing 是否如預期退化？

否證條件： 池化在目標工作負載中始終被本地 DRAM 或儲存方案支配。

SECTION

H6：系列協同

TCE＋O-Chip＋SynCore＋DEO／DPCPS／DryCore 是否在端到端基準中優於各子系統獨立局部控制？

否證條件： 協同層造成控制震盪、過高通訊或無法重現的模型耦合。

SECTION

E0：構想與形式模型

本文目前屬 E0。

SECTION

E1：可執行軟體原型

具備 TopoIR、至少一個重寫後端和 DDLE 基線。

SECTION

E2：可重現單機實驗

公開程式、硬體、資料、編譯參數與完整日誌。

SECTION

E3：多後端整合

CPU、GPU、NUMA／CXL 模擬與至少一個可重構後端。

SECTION

E4：硬體在迴路與第三方驗證

外部團隊可重現主要結論，並公開失敗案例。

SECTION

E5：部署資格化

長期穩定性、安全、版本治理、回退、故障注入與實際工作負載證據。

任何 10^3 或 10□ 倍主張都必須標明：

- 相對哪個基線；
- 是否算法改變；
- 是否精度改變；

- 是否只計算 kernel；
- 是否排除編譯、資料轉換和 I/O；
- 是否來自合成情境；
- 是否可被第三方重現。

SECTION

15.1 計算與幾何的弱命題

合理的弱命題是：

> 計算的表示、資料依賴、通訊與硬體配置具有幾何與拓撲結構；選擇不同表示，會改變可利用的局部性、並行性與驗證方式。

這不等於：

> 計算的本體只剩幾何，或提高維度即可消除複雜度。

SECTION

15.2 「插隊」的真正意義

合法的插隊不是跨越因果，而是：

- 不按來源程式的表面順序執行獨立節點；
- 不計算與當前查詢無關的節點；
- 用符號表示取代不必要的枚舉；
- 用索引、快取、增量和前沿避開重複工作；
- 在合法誤差內使用近似或多保真模型。

因此：

> 跳過表面順序
> ≠
> 跳過真實依賴.

SECTION

15.3 AI 是搜尋協作者，不是超越證明的主體

AI 可以幫助人類在龐大重寫與映射空間中搜尋，但其輸出仍是：

> 候選計畫
> +
> 信心
> +
> 證據需求,

而不是「直接看見概念空間真理」。

SECTION

15.4 CXL 是資源邊界的重畫，不是維度解放

CXL 的重要性在於讓部分記憶體與裝置資源不再被單一主機板插槽永久綁定；它重新配置了系統資源邊界。但所有邊界重畫都伴隨協議、交換、延遲、擁塞、故障與治理。

SECTION

15.5 新版最終命題

原稿的句子是：

> 計算即存在，幾何即命運。

新版提出：

> 表示塑造可見的解空間，拓撲限制可行的執行路徑，證據決定哪些變換能被信任。

形式上：

> 語義
> xrightarrow表示
> 候選拓撲
> xrightarrow約束與成本
> 執行計畫
> xrightarrow驗證
> 可接受結果.

TCE 的價值不在於宣布「後圖靈時代」已到來，而在於建立一個可以逐項實作、量測、否證與迭代的接口：讓程式重寫、查詢局部性、資源拓撲和異質執行不再是互不相干的最佳化孤島。

拓撲約束執行引擎 v2.0 將原稿的無限維計算流形、插隊計算、AI 塑形與 CXL_∞，重構為四個可落地的研究模組：

- TopoIR 與等價重寫空間：以多層圖和 e-graph 表示可選程式結構；
- 需求導向躍遷執行：只計算查詢閉包、稀疏前沿和受變更影響的區域；
- 拓撲感知異質映射：共同最佳化資料、任務、互連、記憶體、功率與故障域；
- 證據化 AI 策略與安全回退：AI 提出候選，形式規則、量測與硬體護欄決定能否執行。

  這些機制不會普遍把 O(n) 變成 O(1)，也不需要假設無限維物理或超圖靈計算。它們真正嘗試解決的是更現實的問題：現代系統明明擁有多種等價表示和多種異質資源，卻經常因固定 pass、全圖執行、錯誤資料放置與局部控制衝突而浪費工作。

  本文目前只是一份 E0 架構提案。下一步不是再增加本體論層級，而是實作 TopoIR、建立 DDLE 反例測試、量測重寫搜尋成本、公開拓撲映射基準，並讓所有「更快」都能被完整成本和第三方重現所檢查。

- MLIR Project, “Multi-Level Intermediate Representation Overview,” LLVM Project. <https://mlir.llvm.org/>
- MLIR Project, “Transform Dialect,” LLVM Project. <https://mlir.llvm.org/docs/Dialects/Transform/>

- MLIR Project, “Dialect Conversion,” LLVM Project. <https://mlir.llvm.org/docs/DialectConversion/>
- Max Willsey et al., “egg: Fast and Extensible Equality Saturation,” POPL 2021. <https://doi.org/10.1145/3434304>
- Fredrik Kjolstad et al., “The Tensor Algebra Compiler,” OOPSLA 2017. <https://doi.org/10.1145/3133901>
- GraphBLAS Forum, “GraphBLAS Specification and Resources.” <https://graphblas.org/>
- Jeremy Kepner et al., “Mathematical Foundations of the GraphBLAS,” 2016. <https://arxiv.org/abs/1606.05790>
- R. Hoque et al., “Dynamic Task Discovery in PaRSEC: A Data-Flow Task-Based Runtime,” 2017. <https://icl.utk.edu/files/publications/2017/icl-utk-1027-2017.pdf>
- Compute Express Link Consortium, “CXL 4.0 Specification Now Available,” 2025. <https://computeexpresslink.org/>
- Compute Express Link Consortium, “Introducing the CXL 4.0 Specification,” 2025. <https://computeexpresslink.org/event/introducing-the-cxl-4-0-specification/>
- Carl Yang, Aydın Buluç, John D. Owens, “GraphBLAST: A High-Performance Linear Algebra-based Graph Framework on the GPU,” 2019. <https://arxiv.org/abs/1908.01407>
- Stephen Chou, Fredrik Kjolstad, Saman Amarasinghe, “Format Abstraction for Sparse Tensor Algebra Compilers,” 2018. <https://arxiv.org/abs/1804.10112>

符號	意義
cal-G_W	工作負載多關係圖
cal-G_R	資源圖
cal-G_E	證據圖
E_D	資料依賴邊
E_C	控制依賴邊
E_M	記憶體與一致性邊

E_S 副作用與順序邊

Π 執行計畫

ρ 等價重寫選擇

μ 任務—資源映射

λ 資料格式與位置

σ 排程與前沿策略

δ 運作包絡與功率—熱要求

ϕ 驗證、失敗與回退計畫

$A(Q)$ 查詢 Q 的祖先閉包

$D(\Delta)$ 變更 Δ 的後代閉包

DDLE 需求導向躍遷執行

$C_{(reconf)}$ 重組總成本

$U_{(model)}$ 模型不確定性

任何 TCE 實驗至少公開：

- 完整工作負載和資料版本；
- TopoIR 或可重建的圖表示；
- 重寫規則與信任等級；
- 編譯、搜尋和抽取預算；
- 資源圖、硬體、驅動和韌體；
- 基線與所有參數；
- 牆鐘時間、編譯時間、能耗與資料搬移；
- 快取、索引、物化和失效成本；
- 失敗、回退與負結果；

- 結果雜湊與可重現腳本。