大數據和 AI 應用的新運算架構

20190518 人工智慧論壇 心得整理- 徐聖智 Steven http://steven.spbc.com.tw

當今的多核心處理技術和架構在性能和效率方面已無法滿足需要即時運算的軟體演算法需求。

大多數運算技術專家都坦言，全球資料量每兩年成長一倍。預計到 2020 年，針對用於人工智慧(AI)相關的先進應用(如機器學習、機器人、自動駕駛和分析以及金融市場)，資料集將超過 21ZB(zettabyte)或更多。

這些類型的應用都依賴於大數據(Big Data)——這需要一些即時計算非常小、但多層級的軟體演算法。然而，如今的多核心處理技術和架構在性能和效率方面已無法有效地滿足這些演算法的需要。

問題：「所有的性能都去哪了？」

直到 2000 年代初期，電腦性能與處理器的時脈頻率相關，根據摩爾定律(Moore's Law)，頻率每 18 個月會增加一倍。到了 2005 年後，摩爾定律顯示，處理器核心數約每隔 18 個月增加一倍。傳統觀點認為，運算性能的進步將來自於向傳統處理器添加更多的核心。然而，如圖 1 所示，最終結果卻發生了邊際效益遞減的情況。

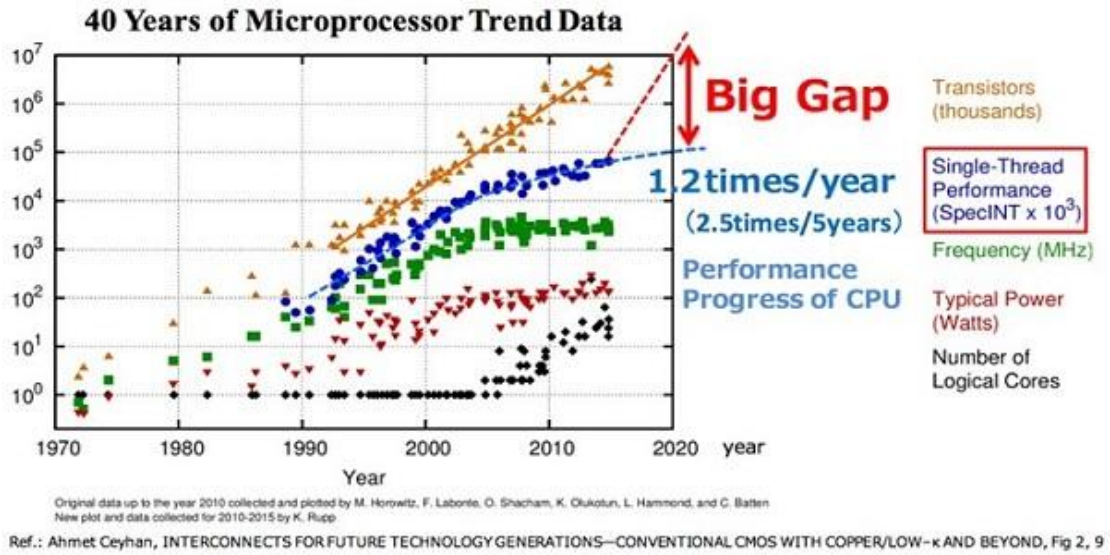

圖 1：多核心 CPU 性能不再持續上升

為了提高傳統軟體演算法的性能而添加越多核心，增量增益就越小(除了一些值得注意的例外)。因此，核心數量止步不前。讓每一代的電晶體數量倍增，使得「少量」核心的速度稍快些，而不是用於增加更多的核心。因此，

電晶體資源的指數級增加被浪費了，而只為我們帶來性能上的小小線性增益，如圖 2 所示。

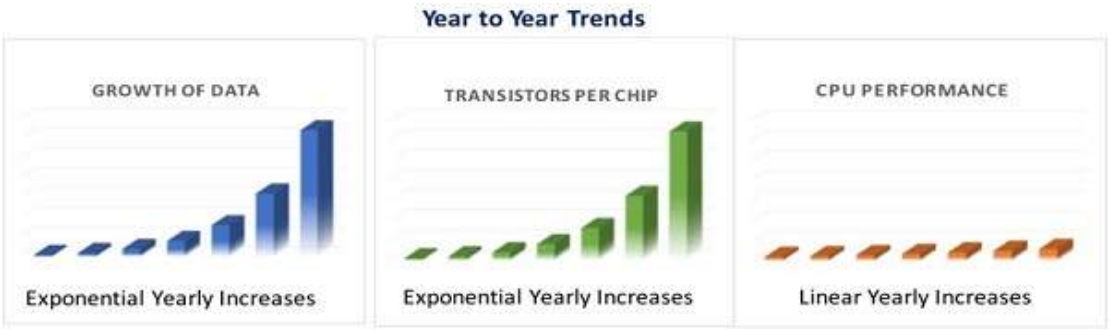

圖 2：大數據和機器學習的處理需求，要求我們更有效地使用電晶體預算

需要處理的資料量正以指數級速度增加。目前這一代「雲端」是由充滿「少量核心」伺服器的大量機架組成。其結果是大量的網路伺服器造成了大規模的資料中心蔓延，從而導致全球用電量和碳排放量大幅增加。

關鍵的是，對於大數據和機器學習工作負載的檢查顯示，它們與現有處理器數十年來設計和最佳化的工作負載大不相同。大數據/機器學習程式碼大小以 kLOC(千行程式碼)為單位，而傳統軟體(例如你最喜愛的辦公室套件或作業系統等)是以 mLOC(百萬行程式碼)為單位進行度量的。例如，中國科學院(CAS)為 SPARK 應用架構提供 BigDataBench 3.2 上的簡單 LOC grep，就涵蓋了十幾個不同的基準測試，顯示總計不超過 1,000 行 SCALA 程式碼的累計 kLOC。Yahoo! Streaming Benchmark 則少於 300 行 SCALA 程式碼。Google TensorFlow MNIST 的整個教程有多個副本，涵蓋超過 1,000 行 python 程式碼。

因此，我們需要一種適合目前工作負載的新處理架構，以提供一種可擴展、大規模平行、海量核心(sea-of-cores)的方法。讓我們使用電晶體預算，透過添加大量核心來提供更多運算，而非僅試圖加速一些核心。

例如，CORNAMI 正致力於克服這一處理與運算性能議題，利用同步技術開發出專利的新運算架構，能夠獨特地改變軟體性能並降低功耗、延遲和平台尺寸。

其結果是在每個處理核心都具有獨立決策能力的大規模平行架構，穿插著高速記憶體，以及所有透過生物啟發的網路互連以產生可擴展的大量核心。該架構基於 CORNAMI 開發的獨特結構，稱為 TruStream 運算結構(TSCF)，可跨多個晶片、電路板和機架進行擴展，每個核心可獨立編程設計。

透過使用 TruStream 程式設計模型(TSPM)，多核心處理器資源被抽離至共同的同質核心池。TruStream 建置於軟體和硬體上，並可在 TruStream 運算結構中運行。程式設計人員可透過嵌入更高層標準語言的 CORNAMI TruStream 控制結構，輕鬆實現同步。

機器學習處理引擎的核心

最近，微軟(Microsoft)的卷積神經網路(CNN)加速器、Google 用於神經網路加速的張量處理單元(TPU)，以及 NVIDIA 關於深度學習 GPU 的 Volta 等消息發佈，都揭示了加速機器學習演算法的類似方法。

要加速的演算法都是 3D 濾波器的變異版本(圖 3)。最消耗 CPU/GPU 處理能力或晶片面積的演算法是 3D 卷積濾波器——而這也正是 CNN 的核心和靈魂。

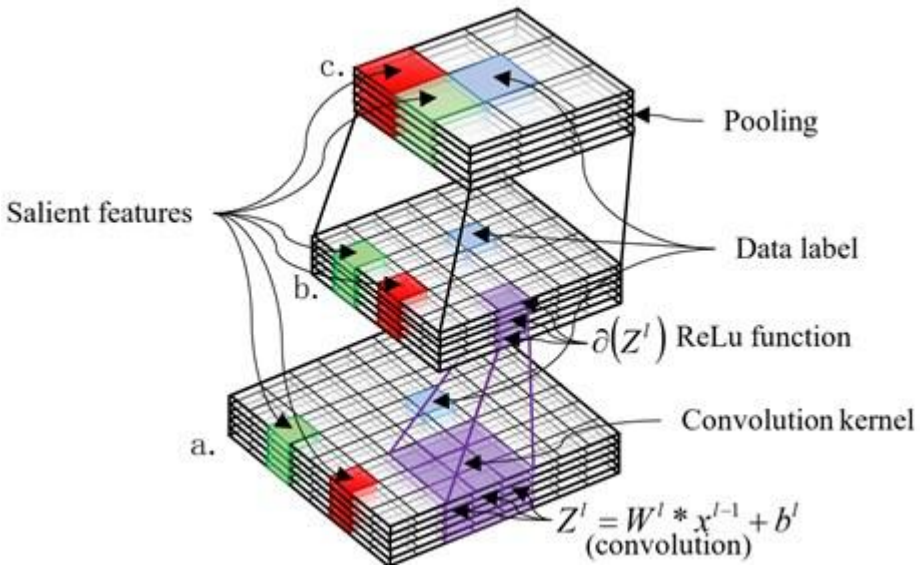

圖 3：卷積神經網路(CNN)顯示 3D Convolution + 3D ReLu + 3D Pooling（來源：《應用科學》，2017,7,447）

為了加速這些演算法，可使用 2D 矽結構——這是一種稱為脈動陣列(systolic arrays)的矽加速器。圖 4 顯示相乘的兩個簡單 3×3 矩陣；圖 5 顯示將執行乘法的 3×3 乘法累加單元脈動陣列。

$$\begin{matrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{matrix} \times \begin{matrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{matrix} = \begin{matrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{matrix}$$

$$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21} + A_{13} \times B_{31}$$
$$C_{21} = A_{21} \times B_{11} + A_{22} \times B_{21} + A_{23} \times B_{31}$$
$$C_{31} = A_{31} \times B_{11} + A_{32} \times B_{21} + A_{33} \times B_{31}$$
$$\vdots \quad \vdots \quad \vdots$$

圖 4：3x3 矩陣乘法

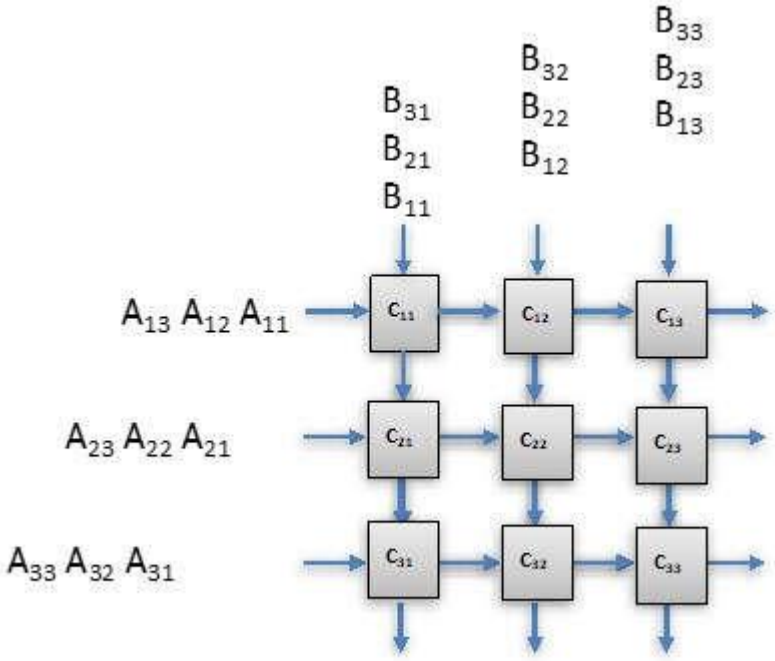

圖 5：執行串流矩陣乘法和累加的 3x3 元素脈動陣列

例如，輸出矩陣元素 C11 包含在 A 和 B 輸入資料完成串流之後的結果 A11·B11 + A12·B21 + A13·B3。較大維數的陣列允許發生大量平行作業；陣列元素之間更複雜的互連增加了頻寬；每個元素的更複雜功能允許執行卷積和其它功能；而更複雜的控制電路允許結果串流輸出，而更多的資料串流輸入且執行更多的運算。

這些是用於生產元件的技術。消除複雜性及核心思想很簡單。請注意，輸入矩陣按照特定的順序串流輸入陣列的左側和頂部，並將結果從陣列中串流輸出。一個脈動陣列的屬性(即能夠連續串流大量資料進出結構；同時操作所有元素；以及中間值保存在脈動陣列內部的事實；而且不需要被「保存」到記憶體中)提供所需的性能。

那麼，如果讓程式設計人員透過軟體來定義任何脈動陣列的維度、功能和互連性呢？也就是說，要保持機器學習所需的高性能，這是脈動陣列的一個副作用；但允許軟體程式設計師透過完全允許在軟體中進行更改，以跟上新演算法的發展。

另外的好處是，這種軟體方法應用範圍相當廣泛，遠遠超出了機器學習的範圍，其中，對 2D 或更高維度陣列的元素(單元、畫素、體積畫素)，及其相鄰元素進行操作。應用範圍包括機器學習、影像處理、壓縮/解壓縮、編碼/解碼、三維(3D)建模、流體動力學、有限元素分析、密碼學、糾錯編碼以及建模實體現實等

最簡單的脈動陣列——The Game of Life

讓我們來看一個前述軟體定義脈動陣列方法的完整實例，該實例使用了 TruStream 程式設計模型，在海量核心 TruStream 運算結構上執行軟體。為了說明將 TruStream 應用到這些問題中的技術，我們選擇了生命遊戲(Game of Life)細胞自動機(cellular automaton)，它由 2D 格狀細胞組成，每個細胞只與以自身為中心的相鄰八格細胞進行互動。幾乎所有的軟體程式設計人員都應該很熟悉這點，並且很容易就可以說明這些機制。當在 TSCF 上運行 Game of Life 的 TSPM 程式時，每個 Game of Life 細胞都執行於其獨特的 TSCF 核心上。

TruStream 程式設計模型

自從亨利福特(Henry Ford)時代以來，組裝脈動陣列的問題與工廠建構師一直在處理的問題非常相似。工廠建構師將問題看作是組織一系列實體(機器、機器人和工人)的問題，這些實體透過從一個實體流向另一個實體的小裝置流進行互動。當這些建構師繪製其工廠圖時，他們繪製的是方塊圖，而不是流程圖。然而，在進行工廠設計時，流程圖的作用是：它們非常適合描述單個機器、機器人和工人的順序行為。總結來說：

確定工人執行任務和佈置工廠的順序是兩個根本不同的活動。

正是這些觀察激發了我們對運算的看法：電腦是組裝資料值的工廠；以及對程式設計的看法：程式的某些部份可最容易且自然地表示為流程圖，程式的其它部份則最容易且自然地表示為方塊圖。或者對等地認為：程式的某些部份在 TSPM 執行緒域中最容易和自然地表達；程式的某些部份在 TSPM 串流域中最容易和自然地表達。

根據這些，我們創建了 TruStream 程式設計模型，該模型基於五個 C++類，如下所示：

threadModule	Encapsulates thread-domain code: -- Sequential code with TrueStream *gets* and *puts*. -- NO parallelism is expressed here.
inputStream<T>	A threadModule input.
outputStream<T>	A threadModule output.
streamModule	Encapsulates stream-domain code: -- Defines a TruStream *topology* (block diagram). -- ALL parallelism is expressed here.
stream<T>	A TruStream -- Connects one or more threadModule outputs to one or more threadModule inputs. -- Appears only inside a streamModule.

圖 6：C++類型構成 C++的 TSPM

為了更直觀地瞭解這些類型，圖 7 和圖 8 提供了五種 TruStream 物件的圖形表示。

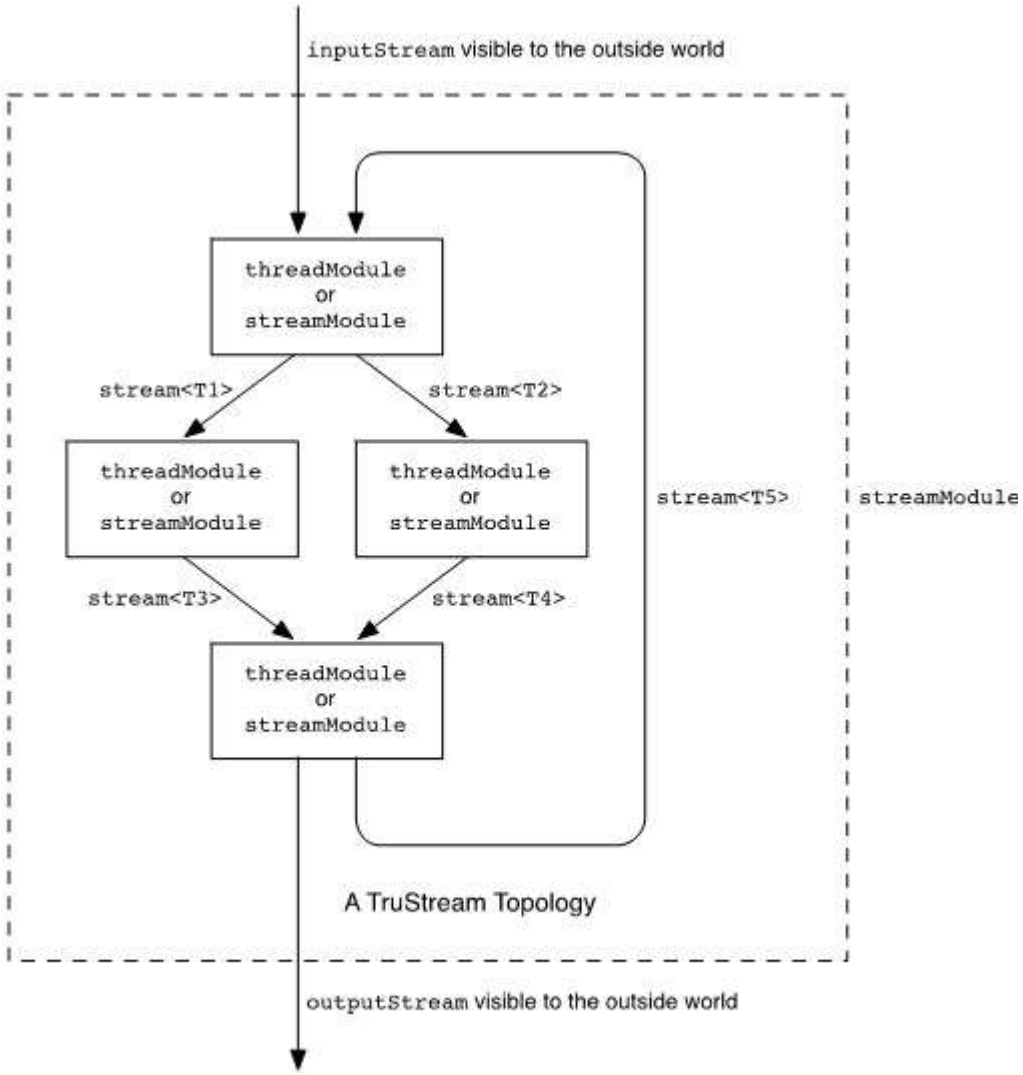

圖 7：TruStream 物件

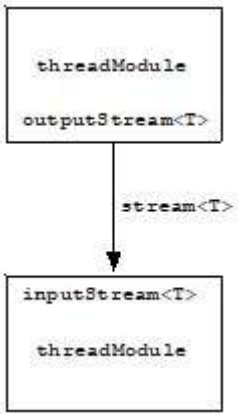

圖 8：輸入和輸出串流

圖 7 顯示 TruStream 程式設計模型的兩個關鍵特性：

- TruStream 拓撲可能是迴圈(或非迴圈)的。

- 透過嵌套 streamModules 創建分層拓撲。

TruStream 程式設計模型有三個獨特屬性：

- TSPM 程式碼分為兩個域：執行緒域和串流域。
- TSPM 僅在串流域中顯示平行性。
- TSPM 依靠單一機制來同步執行緒：TruStreams。

第一個屬性體現了關注點分離的設計原則，讓程式設計師專注於每個域內單獨關注：在執行緒域，程式設計師處理嚴格的順序問題；在串流域，程式設計師處理串流和串流的轉換。

由於這種關注的分離，執行緒域程式設計師和串流域程式設計師都不必擔心多執行緒或將程式碼映射到單個處理器核心。

第二種屬性意味著執行緒域程式設計師(編寫循序程式碼的程式設計師)，不必處理平行性。這與傳統的平行程式設計方法顯著不同。

第三種屬性意味著執行緒域程式設計師和串流域程式設計師都不必處理基於執行緒的平行性結構引起的同步問題(圖 10)。然而，儘管沒有這些結構，TruStream 程式設計師可以存取所有形式的平行性。

透過 Game of Life 例子可知，整個 TSPM 程式由四個 C++類的定義組成，如圖 9 所示：

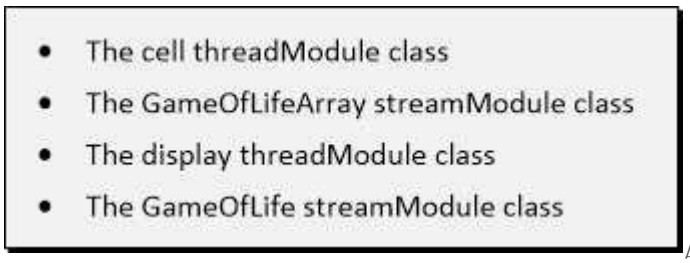

圖 9：構成 Game of Life 的類別（來源：CORNAMI）

圖 10：TruStream 程式設計模型中不包括的結構

TruStream 運算結構

TruStream 運算結構是一個可擴展的海量核心處理器，包含處理核心和晶片上 SRAM。所有核心都可獨立程式設計，並且可平行進行獨立決策(與現代 GPU 不同)，而且該架構可以跨越多個晶片、主機板和機架進行擴展。如同大多數程式設計師不會過度關注其使用的 RAM 數量一樣，TruStream 程式設計師也不必過於關心其所使用的核心數量。

TSCF 和 TSPM 之間的連接包含在以下三個屬性中：

- 在 TruStream 程式中的每一個 threadModule 都有其各自的 TruStream 運算結構核心。
- 在 TruStream 運算結構核心中執行的唯一程式碼是封裝在 threadModule 的執行緒功能程式碼。
- 在 TruStream 程式中的每一個 streamModule 在 TruStream 運算結構中配置一個 TruStream 拓撲。

TruStream 運算結構的主要特性如下：

- TSCF 針對 TruStream 程式設計模型進行最佳化，因此 TSCF(如 TSPM)支援所有形式的平行性。
- TSCF 對於 TruStreams 提供硬體支援，對 TruStream 提供 gets 和 puts 支援，使 TSCF 能夠非常高效地執行 TSPM 程式。
- TSCF 中唯一運行的程式碼是應用程式碼！僅此而已。並未列出結構中運行的 11 個膨脹程式碼(bloat code)。

- TruStream 運算結構中的執行緒等待輸入資料和輸出空間。除此之外，執行緒不會等待。
- 每個 TSCF 核心最多支援一個執行緒，因此不存在上下文切換。這意味著不需要調度或分發程式。
- TSCF 性能來自平行性，而不是蠻力的速度，因此我們有最佳化 TSCF 的能源效率的優勢。
- 由於晶片外(off-die)操作是性能殺手，TSCF 提供大量晶片上 SRAM。
- 每個 TSCF 核心都有自己的記憶體埠，因此 TSCF 提供特別(高)的聚合記憶體頻寬。

總之，這些特性帶來了：(1)延遲顯著降低；(2)性能顯著提高；以及(3)運算能力明顯增加。

圖 11：不包括在 TruStream 運算結構中執行的程式碼

結語

TruStream 技術重新定義了運算的含義。在過去的 70 年中，一直以執行緒為中心的觀點已一去不返。取而代之的是這樣一個模型：執行緒仍扮演一個角色，但是個支援的角色(可以完全消除執行緒；例如，用純功能語言編寫的功能程式碼替換循序執行緒功能變數程式碼，以及用堆疊機替換馮諾依曼核心)。

該技術最顯著的特點是：(a)將程式碼分成兩個域：執行緒域和串流域；(b)僅在串流域中表現平行性；以及(c)依賴單一機制(TruStreams)用於同步執行緒。

這些功能有很多好處，其中主要有以下幾點：

- 在軟體內快速建構任意多維脈動陣列並使其於核心運算結構上執行的能力。你可以實現當今最喜歡的機器學習演算法，而且當更好的機器學習演算法出現時，就取而代之。當系統工作負載發生變化時，軟體可以即時更改脈動陣列的大小和性質。需要結合機器學習功能執行複雜和任意的控制功能嗎？編碼吧！我們無論怎樣強調軟體靈活性的優勢都不為過。利用機器學習所用的高性能矽加速器，你會被 ASIC 流片時制訂的設計決策所束縛，而且矽加速器仍然需要存取通用處理器核心，以便控制決策。
- 可任意擴展的多核心架構。大多數多核心方法都有一個集中化資源(例如調度程式或分發程式)作為限制系統規模的瓶頸。在 TruStream 運算結構中，沒有這樣的瓶頸。一旦 TruStream 拓撲下載到 TSCF 並開始執行，拓撲中的每個執行緒模組都是獨立的。它從輸入串流獲取資料值、執行運算、並將資料值放入輸出串流中。僅此而已。沒有執行時間調度或分發，而且沒有同步原語——超越 TruStreams。此外，由於每個執行緒模組都有自己的核心，所以沒有上下文切換。多個 IC 可以連接在一起建構更大的運算結構；機架中的多個伺服器可以連接在一起；多個機架還可以結合在一起處理更大的工作負載。
- 對於大數據和機器學習應用，優勢轉向擁有越來越多更小核心，而不是試圖盡力提高大型傳統處理器核心的性能。技術節點的變化通常使你可將放在同一尺寸晶片上的電晶體數量加倍。在今天的技術節點實務中，這意味著你會得到另外幾十億個電晶體。傳統方法將利用這些電晶體並將其轉化為 15%至 30%的性能增益。現在，你可以將它們轉換為額外的 4,000 個處理器核心