When Data Movement Becomes the Bottleneck in Modern Workloads: Compute-in-Transit as an Architectural Model

Flavio Bergamaschi
Optalysys Ltd
flavio.bergamaschi [at] optalysys [dot] com

Abstract

In modern computing workloads, performance is increasingly constrained not by computation, but by the cost of moving data. This shift reflects both the scale and structure of contemporary applications, in which large data sets are subjected to repeated transformations across memory hierarchies, interconnects and distributed systems. A similar pattern appears across domains including fully homomorphic encryption, post-quantum cryptography and artificial intelligence: intermediate representations are repeatedly transformed and exchanged, and their movement rather than the arithmetic itself is what governs system efficiency.

This paper examines Compute-in-Transit as an architectural model in which computation is applied during data movement, embedding transformations along the data path rather than at discrete processing nodes. Rather than treating communication and computation as separate processes, this model aligns computation with dataflow, reducing the need for intermediate storage and repeated transfers. While the underlying idea has been explored in prior work, its practical realisation has been constrained by electronic architectures. Photonics provides a distinct approach, enabling transformations to be performed directly on signals in transit and offering a path toward systems in which computation is applied as data moves rather than after it is transported.

1 Introduction

In modern computing workloads, performance is increasingly constrained not by computation, but by the cost of moving data. Despite successive advances in processor performance and memory systems, emerging workloads have revealed a broader system-level constraint, where performance is often determined less by how quickly data can be processed, and more by whether it must be moved at all.

This constraint reflects both the scale of modern data and the structured nature of contemporary workloads, where large data sets are repeatedly transformed across distributed systems.

For decades, computing performance has been understood in terms of successive limiting factors, with early systems constrained by processor throughput and later architectures encountering the memory wall [1, 2] as improvements in computation outpaced data supply.

This progression has now extended further, with a system-level limitation emerging in which the cost of moving data increasingly governs performance, even as computation continues to improve. The energy cost associated with data movement has further reinforced this constraint [3].

Across an increasing range of workloads, performance depends less on computation and more on the cost of transporting data between processing stages. This constraint extends beyond memory access to the wider system, including memory hierarchies, interconnects and increasingly distributed infrastructure. Recent analyses of fully homomorphic encryption workloads have shown that arithmetic intensity can fall below one operation per byte, rendering such systems fundamentally memory-bound rather than compute-bound [4].

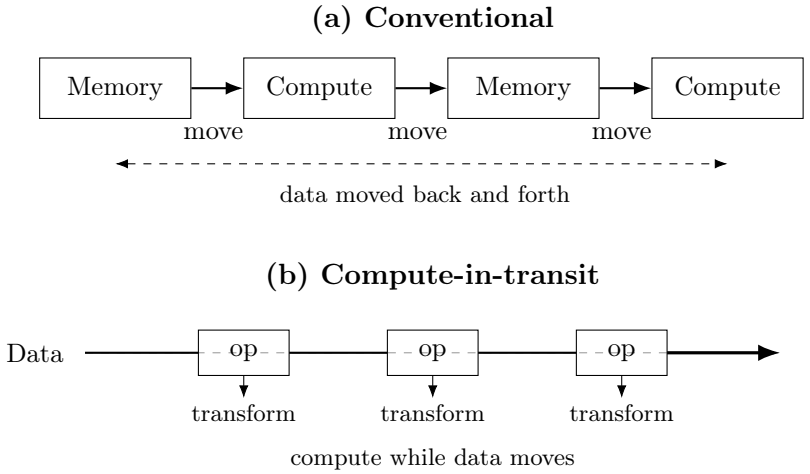

Figure 1: Conventional systems move data between memory and computation, while Compute-in-Transit applies transformations along the data path.

The effect becomes more pronounced as workloads scale and become more structured, with the same operations applied repeatedly to large data sets across distributed systems, leading to increasing movement of intermediate data between processing stages.

The dominant cost is often no longer computation, but the movement of data, a trend already evident across areas such as cryptography, artificial intelligence and machine learning, and large-scale distributed systems. This observation has also been articulated in recent architectural discussions, where emerging workloads have been characterised as constrained by data movement rather than computation [5]. This shift is illustrated in Figure 1, contrasting conventional data movement with computation applied along the data path.

2 Workloads That Are Becoming Data Movement Bound

The growing importance of data movement is evident across a broad class of workloads with similar structural characteristics.

Fully homomorphic encryption operates on large ciphertexts and relies on repeated structured mathematical transforms [6, 7]. While these computations are complex, it is the movement of data between memory and processing units, rather than the arithmetic itself, that has become the primary constraint [4]. In practice, evaluation keys can reach hundreds of megabytes, and key switching operations are dominated by memory bandwidth rather than arithmetic cost [4]. Although ciphertexts must preserve full information, these representations often grow in size and require repeated reduction during processing.

Post-quantum cryptographic schemes exhibit similar behaviour, with lattice-based constructions introducing larger keys and ciphertexts than classical approaches while requiring regular arithmetic patterns at scale [8, 9]. Although the magnitude of expansion is smaller than in fully homomorphic encryption, intermediate representations can grow during polynomial arithmetic and transform operations and require repeated reduction through modular operations. Consequently, data movement and memory bandwidth increasingly govern performance as these schemes are deployed more widely.

Artificial intelligence systems exhibit similar patterns, where workloads involve repeated, regular operations applied to large structured data sets. Intermediate representations, such as large tensors, are propagated across multiple processing stages and distributed components, often requiring

frequent exchange between accelerators and across nodes [10–13]. As model sizes and system scales increase, this communication becomes a dominant factor in system performance, with communication overhead increasingly governing execution time in large-scale distributed training settings [14].

Across these examples, the same pattern repeats: intermediate representations are repeatedly transformed, expanded or exchanged, and it is their movement between processing stages, rather than the arithmetic itself, that governs overall system performance. Despite differences in domain, these workloads share a common structure: large volumes of data are subjected to repeated transformations, making data movement, rather than computation, the dominant constraint.

3 Limits of Conventional Optimisation

With workloads increasingly dominated by data movement, substantial progress has been made in optimising computation and data movement within digital systems, with specialised accelerators improving arithmetic efficiency and advances in memory hierarchies and interconnects reducing latency and increasing bandwidth [1, 3].

However, these approaches rely on a common assumption: computation and data movement remain distinct processes. Data must still be transported to discrete processing elements, traversing memory hierarchies and communication layers before operations can be applied. While the cost of this movement has been reduced, it has not been eliminated.

As workloads scale, this separation becomes increasingly limiting, since faster computation yields diminishing returns when the latency and energy costs of data movement dominate performance. Even in the presence of specialised hardware, improvements in arithmetic throughput alone do not resolve this constraint, as systems remain fundamentally limited by the cost of moving data rather than computing on it [4]. This limitation points to the need for alternative architectural models in which computation is more closely aligned with data movement, rather than remaining separated from it.

4 Compute-in-Transit as an Architectural Model

Compute-in-Transit is one such architectural model, in which computation is more closely aligned with data movement. The concept has been explored in prior work [15–17], but its practical realisation has been constrained by the limitations of existing computing systems.

It is a model in which computation is applied during data movement, embedding transformations along the data path, rather than applying them at discrete processing nodes. This differs from near-memory or in-network computing in that computation is not simply relocated closer to data, but is instead embedded directly within the data path. The emergence of data movement bound workloads has led to renewed interest in reconsidering the separation between computation and communication.

In conventional architectures, data is repeatedly moved across multiple layers of the system, from storage to memory, cache, registers and finally compute elements, before returning along that same path. Each stage introduces latency and energy cost, and as data volumes have grown, this movement has increasingly dominated overall system performance.

Within the Compute-in-Transit model, operations are applied directly to data streams as they propagate, embedding computation within the dataflow and reducing the need for intermediate buffering and repeated memory transfers. The Compute-in-Transit perspective aligns with observations that structured, streamable workloads are often deterministic, enabling computation to be scheduled along dataflow paths rather than tied to fixed processing locations [4].

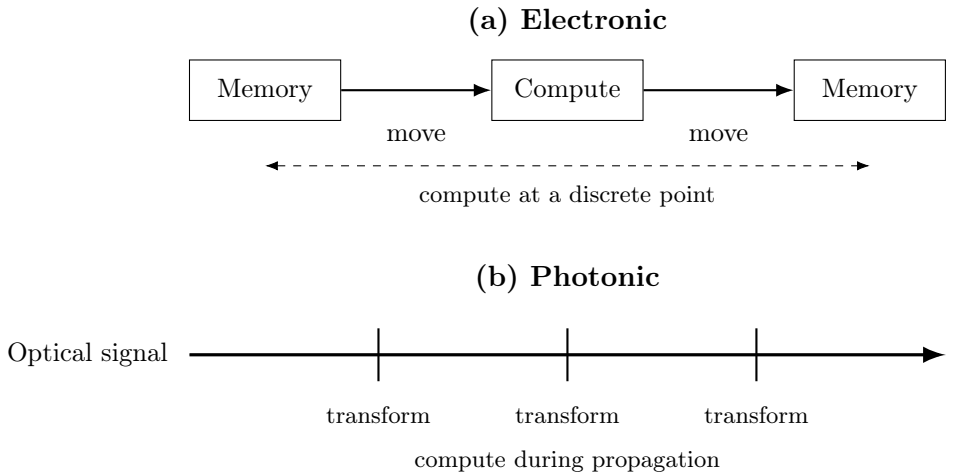

Figure 2: Electronic systems perform computation at discrete processing elements, while photonic systems apply transformations directly to signals in transit along the data path.

This requires an execution model in which computation is expressed as structured, deterministic dataflow, enabling operations to be scheduled and optimised jointly with data movement through intermediate representations and domain-specific instruction sets [18].

Prior approaches have introduced computation within networks by embedding logic into intermediate nodes such as routers or switches [16, 17]; however, these systems still rely on discrete digital processing and do not eliminate the need to move data to points where computation ultimately occurs.

This reflects a broader shift in system design. Rather than optimising the delivery of data to computation, the objective becomes to apply computation along the path of data movement.

Although the initial benefits are primarily observed at the system level, they extend across scales, from system architectures down to the system-on-chip level. Reducing intermediate buffering and repeated transfers lowers latency and energy consumption, thereby reducing pressure on memory bandwidth and interconnects. Performance becomes more closely aligned with data flow rather than the location of compute.

This approach applies naturally to structured, deterministic and streamable workloads, although its applicability depends on the ability to express computation in a streaming form.

A key difference emerges when compared with approaches such as streaming architectures, near-memory computing and in-network processing. While these approaches are often applied to similar classes of structured workloads, they primarily reduce the cost of data movement and still treat communication and computation largely as separate processes. Compute-in-Transit instead embeds computation within data movement itself, aligning transformation with propagation rather than with fixed processing locations. The goal is not simply to minimise data movement, but to exploit it as part of the computation.

5 Photonics as an Enabling Technology

Realising Compute-in-Transit requires technologies capable of operating directly on data as it propagates, and photonics provides such a capability. The distinction between electronic and photonic data paths is shown in Figure 2.

In these systems, information is carried by optical signals rather than electrical currents, enabling

high-bandwidth transmission across multiple wavelengths [19, 20]. Optical propagation through a medium supporting interference and diffraction inherently implements linear transformations, allowing operations such as Fourier transforms and convolutions to be applied directly to signals in transit, without intermediate storage or scheduling [21–23].

In practice, optical interference between coherent light beams produces Fourier-domain transformations as the signal propagates, effectively performing computation without discrete execution steps [24].

These transformations map naturally to polynomial operations, including FFT, NTT and modular arithmetic, that underlie FHE workloads, allowing core primitives to be executed at the physical layer [24].

These operations are particularly effective for structured linear transformations, although general-purpose computation still requires electronic processing. Thus, computation becomes a property of the transmission process itself, effectively embedding the operation within the propagation medium.

In this context, photonic systems enable a unification of computation and communication, in which operations are performed directly on optical data streams, effectively collapsing the distinction between data movement and computation [25].

Such operations are inherently wide and shallow, applying high-dimensional transformations in parallel across entire optical wavefronts in a single step.

This contrasts with electronic systems, where computation is realised through sequences of discrete operations, and allows photonic systems to align computational throughput with data bandwidth.

Photonics also enables parallelism across wavelengths and channels, which allows computational capacity to scale with data bandwidth rather than with the number of discrete processing units. The implications are architectural, affecting latency, energy consumption and scalability, as reducing data transfers and intermediate storage lowers system cost and eases bandwidth constraints.

These approaches complement conventional digital computing. Electronic systems remain essential for general-purpose processing, control and nonlinear operations, while photonic systems offer advantages for structured, data-intensive workloads that can be expressed in streaming form, enabling computation to be applied directly during data propagation rather than after transport.

6 The Emergence of Data Movement Bound Architectures

A consistent architectural shift is emerging across domains, from fully homomorphic encryption to post-quantum cryptography, and from artificial intelligence systems to distributed computing, in which the same constraint has repeatedly appeared: it is the cost of moving data, rather than computing on it, that has become the dominant factor.

This shift represents a transition from compute-bound systems to data movement bound systems, where the constraint is no longer confined to memory access but extends across the full data path, from physical propagation through interconnects to system-level data movement, as data volumes have grown and computations have become more structured. System performance is increasingly constrained by the movement of data across memory hierarchies and interconnects, a shift widely recognised in modern computer architecture [1, 5].

Historically, computing systems have been constrained by limitations in computation and memory access; the emerging constraint is more fundamental, as it concerns not only how efficiently data can be accessed, but whether it must be moved at all.

Under these conditions, communication and computation are no longer distinct stages, but aspects of the same physical process, suggesting that Compute-in-Transit represents not merely an

optimisation of existing systems, but a shift in how computation is realised. Rather than moving data to where computation occurs, computation becomes a property of how data moves, and architectures that align performance with data flow are therefore likely to define the next phase of computing.

Acknowledgements

I would like to thank colleagues for thoughtful discussions and in particular those who challenged some of the assumptions behind this work.

References

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 2019.

[2] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, 1995.

[3] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *IEEE International Solid-State Circuits Conference*, 2014.

[4] C. Gentry, "Practical fully homomorphic encryption: Three questions for the architecture community," 2026, invited talk, ASPLOS 2026 (unpublished).

[5] F. Bergamaschi, I. Kundu, J. Wilson, J. Crawford, and F. Michel, "Hardware-software co-design for advanced hardware accelerators," 2026, tutorial, CPA@ASPLOS 2026.

[6] C. Gentry, "A fully homomorphic encryption scheme," in *STOC*, 2009.

[7] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT*, 2017.

[8] J. W. Bos *et al.*, "Post-quantum cryptography," in *NIST Workshop*, 2015.

[9] National Institute of Standards and Technology, "Post-quantum cryptography standardization," 2024, https://csrc.nist.gov/projects/post-quantum-cryptography.

[10] V. Sze, Y.-H. Chen, T.-J. Yang, and J. Emer, "Efficient processing of deep neural networks," *Proceedings of the IEEE*, 2017.

[11] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*, 2017.

[12] D. Narayanan *et al.*, "Efficient large-scale language model training on gpu clusters," in *SC*, 2021.

[13] Z. Jia, M. Zaharia, and A. Aiken, "Beyond data and model parallelism for deep neural networks," in *SysML*, 2018.

[14] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020.

[15] D. L. Tennenhouse *et al.*, "A survey of active network research," *IEEE Communications Magazine*, 1997.

[16] A. Sapio *et al.*, "In-network computation is a dumb idea whose time has come," in *HotNets*, 2017.

[17] G. Li *et al.*, "Flowblaze: Stateful packet processing in hardware," in *NSDI*, 2019.

[18] I. Kundu and F. Michel, "Memory architecture," UK Patent GB2 617 190, filling date 7 February 2024, publidhng date January 8, 2025. [Online]. Available: https://www.search-for-intellectual-property.service.gov.uk/GB2617190
[19] D. A. B. Miller, "Optical interconnects to silicon," *Proceedings of the IEEE*, 2010.
[20] H. J. Caulfield and S. Dolev, "Why future supercomputing requires optics," *Nature Photonics*, 2010.
[21] E. Cottle, F. Michel, J. Wilson, N. New, and I. Kundu, "Optical convolutional neural networks – combining silicon photonics and fourier optics for computer vision," *arXiv preprint arXiv:2103.09044*, 2020.
[22] Y. Shen *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature Photonics*, 2017.
[23] J. Feldmann *et al.*, "All-optical spiking neural network using phase-change materials," *Nature*, 2019.
[24] I. Kundu, J. Todd, O. Hammond, and P. Simpson, "Dense data packing and transceiver circuit for OFT using integrated photonics," UK Patent GB2 637 934, filling date 7 February 2024, publishing date August 13, 2025. [Online]. Available: https://www.search-for-intellectual-property.service.gov.uk/GB2637934
[25] I. Kundu and R. Todd, "Data transfer and processing," UK Patent GB2 643 955, filling Date 10 September 2024, publishing date March 11, 2026. [Online]. Available: https://www.search-for-intellectual-property.service.gov.uk/GB2643955