International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Balancing Performance and Energy Efficiency: A Comparative Review of Near-Data Processing, Heterogeneous Cores, Persistent Memory Controllers, and Chiplet-Based Disaggregated Memory

Dan Michael C. Carajay, Andrea Rose N. Diaz, Jay Ar P. Esparcia
Department of Computer Engineering
University of Southern Mindanao, Kabacan, Cotabato, Philippines

Abstract
Computers keep getting faster, but moving data between memory and the processor can now cost more time and energy than the computation itself. This paper presents a comparative review of four architectural directions that try to deal with this problem: near-data processing (NDP), heterogeneous core integration, persistent memory controllers, and chiplet-based disaggregated memory. NDP moves part of the work to where the data already sits, so fewer transfers are needed. Heterogeneous processors mix fast cores with frugal ones and give each task to whichever suits it better. Persistent memory controllers manage memory that keeps its contents without power, so they have to balance durability, write endurance, and latency. Chiplet and disaggregated designs break a system into smaller pieces that can be scaled separately, but they bring interconnect delays, coherence problems, and harder software management. Recent survey literature was reviewed and compared in terms of performance, energy efficiency, scalability, reliability, programmability, and cost. The review finds that no single approach wins on every workload. NDP fits data-heavy tasks built from simple operations, heterogeneous cores fit mixed workloads, persistent memory fits applications that need durable state, and chiplets fit systems that have to grow in flexible ways. The most promising path is to combine them through coordinated hardware and software design and to measure energy at the whole-system level. Gaps remain in shared benchmarks, workload-aware scheduling, and cross-layer evaluation.

Index Terms— Computer architecture, Energy efficiency, Heterogeneous computing, Near-data processing, Persistent memory, Chiplets

I. INTRODUCTION

Computer architecture has long relied on faster processors, more parallelism, bigger caches, and wider memory buses to improve performance. That recipe works less well now. Modern workloads such as AI inference, data analytics, cloud services, and scientific simulation touch very large amounts of data, and every trip between memory and the processor spends time and energy that faster arithmetic units cannot win back. The gap between how quickly a processor can compute and how quickly memory can feed it is usually called the memory wall, and it remains one reason why gains in performance-per-watt have slowed down [1], [2].

Researchers have responded in several ways, and four of them are compared here. Near-data processing (NDP) moves computation closer to the data. Heterogeneous core integration puts cores and accelerators with different speed and power on the same chip. Persistent memory controllers manage memory technologies that hold data without power, which means writes, ordering, and recovery must be handled with care [3]. Chiplet-based design splits a processor into smaller dies joined in one package [4], while memory disaggregation separates memory from compute so that each can be provisioned on its own [5].

These ideas share a goal but not the problems they solve, and none of them comes free. A design that cuts data movement can be harder to program. One that improves average energy use can slow certain requests. One that adds memory capacity can also add interconnect overhead. Most studies also look at a single technique at a time, so it is hard to see how the techniques stack up against one another.

This paper compares the four approaches using one shared set of criteria. It looks at what each one is good at, where it struggles, and which research gaps are holding back better performance-per-watt. The aim is to help readers choose a suitable direction for a given workload and to point out where more work is needed.

II. METHODS AND PROCEDURES

This study covers four architectural approaches: near-data processing, heterogeneous core integration, persistent memory controllers, and chiplet-based disaggregated memory. No processor was built or tested. Instead,

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

published surveys and reviews were read and compared using a narrative approach. The goal is to find common benefits, recurring limits, and open questions rather than to rank the designs.

A. *Literature Search Strategy*

The sources were collected from academic and publisher databases such as MDPI, ScienceDirect, IEEE Xplore, and SpringerLink. Peer-reviewed surveys and review articles published mainly between 2022 and 2026 were preferred, since they summarize many individual studies at once. Two older foundational papers, one on the memory wall and one on heterogeneous cores, were kept for background [6], [7]. The search phrases included "near-data processing energy efficiency," "heterogeneous processor architecture," "persistent memory controller," "chiplet-based design," and "disaggregated memory architecture."

A source was included only when it met the following criteria:

1. It must describe or evaluate at least one of the four approaches.
2. It must discuss performance, energy, memory behavior, or system integration.
3. It must explain architecture-level trade-offs clearly enough to be compared.
4. It must be a peer-reviewed article or a survey from a recognized publisher.

Sources were not used as primary evidence if they:

- Were unrelated to processor or memory architecture,
- Gave too little technical detail to support a comparison,
- Were published before 2022, except for foundational works used as background.

This filtering kept the review limited to material that could actually support an architecture-level comparison.

B. *Categorization Framework*

Each approach was treated as its own category, based on the main inefficiency it targets:

1. *Near-Data Processing*
   Computation placed in or near memory to reduce data transfers, including processing-in-memory (PIM) inside the memory device or its logic layer.
2. *Heterogeneous Core Integration*
   Chips that combine high-performance cores, energy-efficient cores, and specialized accelerators, with tasks assigned to whichever fits best.
3. *Persistent Memory Controllers*
   Controllers for non-volatile or hybrid DRAM and NVM systems, which handle request scheduling, write ordering, and recovery.
4. *Chiplet-Based and Disaggregated Memory*
   Systems built from smaller dies in one package, or systems where memory is pooled separately from compute and reached through a fabric.

Chiplets and disaggregation are grouped because both deal with modularity, but they are discussed separately, since one is a packaging choice and the other is a system organization choice.

C. *Evaluation Criteria*

To keep the comparison consistent, each approach was examined along six dimensions:

1. *Performance and Latency*
   Throughput and the delays involved in reaching data.
2. *Energy Efficiency*
   Energy spent on moving data and on keeping hardware active.
3. *Scalability*
   How well compute and memory capacity can grow.
4. *Reliability and Persistence*
   Data durability, error handling, and recovery behavior.
5. *Programmability*
   The compiler, runtime, and operating system support that is needed.
6. *Deployment Cost and Complexity*
   Packaging, integration effort, and hardware cost.

These criteria were chosen because performance-per-watt comes from the whole system and not from the processor core alone.

D. *Comparative Analysis Approach*

Results from the reviewed studies were not combined into one score or ranking. The studies use different workloads, platforms, memory technologies, and ways of measuring power, so putting their numbers side by side would be misleading. A qualitative comparison was used instead. The analysis looks for:

- Recurring benefits of each approach,
- Consistent limitations and costs,
- Conditions under which each approach fits best,
- Gaps in evaluation and software support.

This way each approach is judged in its own context, instead of against figures taken from a different setup.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

III. COMPARATIVE ANALYSIS OF ARCHITECTURAL APPROACHES

This section goes through the four approaches one at a time and then compares them. For each, the focus is on the main benefit, the main limitation, and the kind of workload it suits.

A. Near-Data Processing

Near-data processing (NDP) places processing logic close to memory, and processing-in-memory (PIM) goes a step further by performing selected operations inside the memory device or its logic layer. The motivation is simple: avoid sending every operand across the memory bus to a central processor. The workloads studied most often are neural-network inference, database operations, graph processing, and large-scale analytics [1], [2].

The main strengths of NDP include:

- Less data movement between memory and processor
- Better use of the available memory bandwidth
- Higher throughput and energy efficiency for data-heavy workloads

However, NDP is not useful everywhere. Memory-side compute units usually support only a small set of operations, and moving a task into memory may require changes to compilers, runtimes, data placement, and cache coherence. Thermal limits, memory technology constraints, and the cost of offloading a task also eat into the gains [1], [2].

Because of this, NDP works best when a workload moves a lot of data for each unit of computation and can be written in terms of the supported operations. Programs with irregular control flow or frequent communication will likely see smaller benefits.

B. Heterogeneous Core Integration

Heterogeneous processors put different kinds of compute units on one chip, typically high-performance cores, energy-efficient cores, and specialized accelerators. Early work showed that a single-ISA chip with cores of different sizes could cut power while keeping most of the performance [7]. The operating system, runtime, compiler, or a hardware scheduler then sends latency-sensitive or compute-heavy tasks to the fast cores and background or parallel tasks to the efficient ones. The idea is to stop paying the energy cost of a big core for every task.

The primary advantages of heterogeneous designs include:

- Adaptability to workloads with different demands
- Better performance-per-watt when tasks are matched to the right core
- Accelerators that run selected operations more efficiently than general-purpose cores

Scheduling is the hard part. Task migration, cache locality, synchronization, and differences between core types all add overhead, and the software has to expose enough parallelism for the extra cores to matter. A poor assignment can lower performance or raise energy use instead of improving it.

Heterogeneous cores and NDP complement each other. Compute-heavy tasks can go to performance cores while data-heavy tasks go to memory-side units. For this to work, the controller has to think about where code runs and where data sits at the same time, otherwise it just ends up moving data around for nothing.

C. Persistent Memory Controllers

Persistent memory keeps its contents after power is lost, which helps applications that need durable state, quick recovery, or large memory capacity. The controller sits between the processor and the memory device and coordinates writes, ordering, error handling, and the interaction with caches and software persistence mechanisms. Many designs pair volatile DRAM with non-volatile memory to balance latency, bandwidth, capacity, and persistence [3].

A well-designed controller can improve efficiency through request scheduling, buffering, placement policies, and coordination between memory tiers [3]. The difficulty is that persistence adds requirements that ordinary DRAM controllers do not face. Ordering and recovery have to stay correct, and write endurance and slower writes must be taken into account. Buffering or reordering writes can speed things up, but it cannot be allowed to break durability guarantees.

For this reason, raw bandwidth is not a good enough measure. Persistent memory should also be judged by read and write latency, energy per operation, persistence overhead, endurance effects, recovery correctness, and the cost of coordinating with the operating system and applications.

D. Chiplet-Based and Disaggregated Memory

Chiplet-based design builds a system from several smaller dies placed in one package. Different chiplets can hold cores, cache, I/O, or memory interfaces, so designers can mix process technologies and scale only the parts that need it. Reviews of the field point to modularity and design flexibility as the main benefits, while interconnect design, packaging, partitioning, and cross-chip communication remain the main challenges [4].

Memory disaggregation is related but not the same. Here, compute and memory are separated so that each can be provisioned on its own. A shared pool of memory can be handed out to machines as demand changes, which improves utilization and avoids stranded

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

capacity [5]. The trade-off is that remote memory adds latency, network traffic, congestion, and software complexity, so data placement and locality become central design concerns.

Put simply, chiplets are a physical integration choice and disaggregation is a system organization choice. They can be used together, but they should not be treated as the same thing. Chiplets can supply modular memory controllers or I/O inside a package, while disaggregated memory may link resources across a system fabric. Both need attention to interconnect energy, reliability, bandwidth sharing, and fault management.

E. Cross-Approach Comparison

Putting the four approaches side by side shows the following trends:

1. *Near-Data Processing*
   Reduces data movement and memory traffic, but limited operations and programming and coherence complexity hold it back. It fits data-intensive tasks with reusable or parallel operations.
2. *Heterogeneous Cores*
   Match tasks to suitable performance and power units, but need good scheduling and add software complexity. They fit mixed workloads with different latency and compute needs.
3. *Persistent Memory Controllers*
   Support durable data and hybrid memory management, but write latency, endurance, ordering, and recovery all constrain them. They fit applications that need persistent state or fast recovery.
4. *Chiplets and Disaggregation*
   Offer modularity and flexible compute and memory capacity, but interconnect overhead, packaging, latency, and coordination add cost. They fit scalable systems whose capacity needs keep changing.

Each approach targets a different source of waste. NDP goes after data movement, heterogeneous cores after the mismatch between tasks and compute resources, persistent memory controllers after durable and efficient memory access, and chiplets and disaggregation after modularity and resource utilization. The biggest system-level gains will probably come from combining compatible techniques instead of picking just one.

IV. DISCUSSION

Looking across all four approaches, each one fixes a real problem, but the benefit depends heavily on the workload and on software support. The biggest open issues are how to measure efficiency fairly, how the layers of a system should coordinate, how results can be compared across studies, and what should be researched next.

A. Performance-per-Watt as a System-Level Metric

Performance-per-watt depends on the entire path a workload takes: computation, cache and memory accesses, interconnect transfers, scheduling decisions, and idle power. A faster core does not help much if it keeps waiting for data. In the same way, moving computation near memory can save transfer energy, but it may need extra logic or raise local power density. Evaluations should therefore report throughput and latency together with total system energy, not just processor utilization or peak performance.

B. Software and Cross-Layer Coordination

None of these mechanisms works on hardware alone. NDP needs compiler or runtime support to find work that can be offloaded. Heterogeneous systems need schedulers that know what each core can do. Persistent memory needs correct persistence semantics, and disaggregated memory needs operating-system and application support for allocation and locality. When each layer optimizes by itself, the result can be extra transfers or scheduling overhead. This is why coordinated design across architecture, operating systems, compilers, and applications matters.

C. Benchmarking and Reproducibility

Another problem is that published studies use different benchmarks, technologies, system sizes, and baselines. A result on a memory-heavy neural network says little about a latency-sensitive database or a general-purpose application. Future studies should report workload characteristics, hardware configuration, memory hierarchy, interconnect assumptions, where power was measured, and tail latency. Energy per completed task and results under realistic mixed workloads would also make comparisons more useful.

D. Trade-Off Between Efficiency and Complexity

A recurring tension runs through all four approaches: the more a design saves in energy, the more it usually asks of programmers and system software. NDP and heterogeneous cores need new scheduling and offloading logic, persistent memory needs careful recovery code, and disaggregated memory needs locality-aware placement. For a vendor or a data center operator, a modest efficiency gain may not justify a system that is much harder to program, test, and maintain. Adoption therefore depends on cost and software effort as much as on the efficiency numbers.

E. A Possible Combined Design

A practical future architecture could combine heterogeneous cores with near-memory compute and a tiered memory system. Persistent data would be managed by controller policies that preserve correctness, and chiplet-based parts would give modular integration. Such a design should still be judged as a whole system, since a gain in one component can be wiped out by communication or coordination costs somewhere else.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

F. Emerging Research Directions

Based on the gaps identified, several research directions look worth pursuing:

1. Workload-aware scheduling that considers core selection, memory placement, and NDP offloading together.
2. Persistent memory controller policies that adapt to changing read/write mixes while keeping persistence guarantees.
3. Efficient interconnects, locality-aware placement, and fault isolation for chiplet and disaggregated memory systems.
4. Standardized benchmarks that compare performance, energy, latency, capacity utilization, and reliability under the same conditions.
5. Lightweight management policies and transparent programming models that make these architectures easier to adopt.

Addressing these gaps will be necessary before these architectures can move from research prototypes to widely used systems.

V. CONCLUSION

This paper compared four architectural approaches for improving processor performance and energy efficiency, namely near-data processing, heterogeneous core integration, persistent memory controllers, and chiplet-based disaggregated memory, drawing on recent survey literature. The point was to see what each approach offers and where it runs into trouble.

Near-data processing cuts the cost of moving data, but it supports only certain operations and needs software changes. Heterogeneous cores adapt compute resources to each task, though they depend on good scheduling. Persistent memory controllers handle durable data while balancing latency and correctness, at the price of write limits and more complicated recovery. Chiplet and disaggregated designs add modularity and flexible capacity, but they bring interconnect overhead and coordination problems.

No single technique gave the best result for every workload. Getting good performance-per-watt means matching the architecture to how the workload behaves and measuring the full system, including memory and interconnect energy. Differences in benchmarks and measurement methods across studies also keep direct comparisons from being reliable.

Future work should focus on common benchmarks, scheduling across layers, locality-aware data placement, persistent memory correctness, and scalable interconnects. Combining these techniques carefully is a promising route to computer systems that are both efficient and adaptable.

REFERENCES

[1] M. Hassanpour, M. Riera, and A. González, “A Survey of Near-Data Processing Architectures for Neural Networks,” Machine Learning and Knowledge Extraction, vol. 4, no. 1, pp. 66–102, 2022, doi: 10.3390/make4010004.

[2] K. Asifuzzaman, N. R. Miniskar, A. R. Young, F. Y. Liu, and J. S. Vetter, “A survey on processing-in-memory techniques: Advances and challenges,” Memories—Materials, Devices, Circuits and Systems, vol. 4, art. 100022, 2023, doi: 10.1016/j.memori.2022.100022.

[3] “A survey on hybrid memory systems integrating Non-Volatile Memory (NVM) and DRAM: Architectures, management techniques, and research directions,” Journal of Systems Architecture, vol. 170, art. 103603, Jan. 2026, doi: 10.1016/j.sysarc.2025.103603.

[4] X. Li and S. Yin, “Review of chiplet-based design: system architecture and interconnection,” Science China Information Sciences, 2024, doi: 10.1007/s11432-023-3926-8.

[5] “Disaggregated Memory in the Datacenter: A Survey,” IEEE Access, vol. 11, pp. 20688–20712, Feb. 2023, doi: 10.1109/ACCESS.2023.3250407.

[6] 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, Mar. 1995.

[7] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, “Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction,” in Proc. 36th Annual IEEE/ACM Int. Symp. on Microarchitecture (MICRO-36), 2003, pp. 81–92.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)