Review: "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"

Summary

The paper outlines the memory bandwidth bottleneck of modern computer systems and proposes a solution by introducing a Processing-in-Memory accelerator called **Tesseract**. It outlines its design and functionality, establishes a performance benchmarking methodology, and analyzes the results. On the conceptual side, it makes the following points:

- Modern computer systems are bottlenecked by their memory bandwidth as CPUs are limited by their I/O and power pins.
- As a consequence, to double the memory that can be computed, we need to double the cores, which is inefficient.
- An in-memory processor can provide much higher memory bandwidth, therefore allowing for more efficient scaling.

The paper proposes **Tesseract** with the following design characteristics:

- Tesseract is an in-memory processor that is integrated into DRAM memory.
- Tesseract is made up of an in-order core and a prefetching system with list and message prefetching.
  - *List prefetching* allows for prefetching of predictable stride patterns.
  - *Message prefetching* allows the host CPU to pass hints with the instruction as to what memory should be prefetched.
- The host CPU can offload instructions through either blocking or non-blocking function calls to Tesseract.
  - Blocking function calls block the CPU until a response (a return value) arrives.
  - Non-blocking functions do not need to be waited for, as they have no response.

The paper evaluates the Tesseract system as follows:

- It uses a set of "interesting" problems that are known to be very memory bandwidth-intensive, such as graph traversals. These have poor spatial and temporal locality, resulting in a high number of cache misses.
- The performance of the Tesseract system is compared to a conventional DDR3-based system.
- The Tesseract system vastly outperforms the conventional setup on all tests, illustrating that this improvement is a direct consequence of overcoming the memory bandwidth bottleneck.
- A further analysis of Tesseract's prefetching and barrier/interrupt mechanisms is performed, showing that the long message queue allows the system to perform prefetching very efficiently.

Strengths

The paper clearly illustrates the memory bandwidth bottleneck, proposes a solution, and demonstrates the solution's performance. The shown Tesseract system provides clear advantages in the test cases. It proves that for high-memory-bandwidth computation, an in-memory processor outperforms a conventional processor by a wide margin. The Tesseract system is configurable, allowing application not only in the graph problems proposed in the paper but also in other problems that encounter the memory bandwidth bottleneck.

Weaknesses

The paper only shows performance on a specific set of tests and does not provide results for tests that would not favor or necessitate an in-memory processor. While that is not the main point of the paper, there would be great value in seeing what costs or benefits this setup has on other test cases. This would be important to assess whether it is just a specific solution for a large but still narrow problem, or whether it can serve as a general solution.

In the paper, it is also stated that the system is cost-efficient, but no numbers or further analysis are provided. While that clearly is not the core point of the paper, it is still important to assess not only the technical but also the financial viability to a greater degree than simply stating that it is "cost-efficient," which is an unclear term.

My POV

I see great value in exploring solutions to the memory bandwidth bottleneck. The proposed Tesseract system seems very capable and performs exceedingly

well on the given tests. I would have included a set of tests that would challenge the Tesseract system — for example, highly spatially and temporally local processes would be very interesting. Furthermore, I would be interested in the manufacturing cost of the system. While I understand that engineering cost cannot be fairly assessed for a research project, the manufacturing cost would provide helpful insight into the actual large-scale viability of the solution.

Takeaways

- In-memory processing is a viable way to address the memory bandwidth bottleneck. It provides a scalable solution that does not necessitate increasing the number of CPUs to achieve higher memory throughput.

- Memory prefetching can be very powerful in the right circumstances. If there is ample time between instruction decision and execution, or if there is strong knowledge about the following memory accesses (e.g., strided access patterns), then memory prefetching allows for very high performance.