Review 1: "You and Your Research" by Richard Hamming

Summary

This is a talk by Richard Hamming about the impact of one's relationship with research on the research itself. He talks about the following key points:

- The importance of working on important problems.
- The compound interest of learning and thinking, and the correlation between personal investment in research and the produced results.
- The necessity of working with the system.

He explains that most of the research done is not "important" research. He emphasizes that we need to think about what research we do, how it is seen in the bigger picture, and whether it is important in the bigger picture. In short, he wants researchers to work on topics they find important and that actually further their field.

He explains that time investment, given that the time is invested in a smart way, produces a compound interest. He notes that given two people of identical ability, the one who invests just one hour more per day will produce far better results than the other. He attributes this to the compound interest of learning and thinking — if you spend just a small amount more today, this will lead you to be slightly more productive tomorrow, further increasing the value of the one additional hour you spend. He therefore reasons that if a person wants to produce truly great research, they need to be willing to prioritize their research over other parts of their life to gain this compound advantage.

He also dives into the reality of the system and the institutions that research is confined to. He acknowledges that every good researcher has a certain drive to fight the system, but he conveys that by leaning into the system and working with it, you can exploit its efficiencies without wasting time fighting it. He tells us that this is a necessity even if it comes at the cost of one's individuality and convictions.

He also talks about the importance of scientific dialogue with colleagues — not only from the same field — and the incredible value contained in discussing one's work with capable people.

Strengths

The core strengths of the paper — or rather, the talk — lie in Hamming's experience both within research and in his ability to articulate his ideas. Mainly, the point of working on important problems is a very strong one under the maxim of wanting to achieve the best possible research.

Weaknesses

The main issue with the paper is that it is not reflective on its own ideas. His perception is very much confined to the American system of the 20th century. And while some points with certainty still apply today, it is also necessary to appreciate the importance of context. He also leaves the term "important research" very ambiguous. He connects it closely with accolades and historical relevance, but a great deal of foundational research without much glory is essential to the functioning of not just modern science but also our society as a whole.

My POV

I agree with his points about the compound interest of time investment and the importance of working on "important" problems, but I disagree with his definition of "important." I find that a researcher testing the results of someone's paper and affirming or refuting it is also important. Further, I assume that the system has changed quite a bit, even though I cannot fairly assess that due to a lack of knowledge.

Takeaways

My main takeaways are the following:
In research, you have to be wholeheartedly committed to the work you are doing if you want it to succeed. And if you commit to something, then you should, both beforehand and during, think and reflect on whether what you are doing is important — if not to the world, then at least to yourself. I see this as a point that applies not only to research and science but to almost anything in life.

Review 2: "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.

Review 3: "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"

Summary

The paper outlines the restrictions imposed by the memory bandwidth bottleneck and proposes a solution through bitwise memory operations on entire DRAM lines. It does this in the following steps:

- The paper explains that bulk bitwise memory operations — AND, OR, NOT, etc. — performed on large stretches of contiguous data are very inefficiently handled in conventional systems, as these are simple instructions that must be repeated over large amounts of data and thus become bottlenecked by the memory bandwidth.

- The paper introduces the concept of in-memory bulk bitwise operations. It explains a system using standard DRAM design to utilize simultaneous loading/enabling of memory lines to perform AND and OR operations on entire memory lines. In short: this is possible because the bitlines span orthogonally to the word lines through the DRAM and share a single sense amplifier for all entries. By activating three lines at the same time, the voltage over the three lines is averaged. This gives us a control line **C** and two input lines **A** and **B**, on which we can perform the following operations: `C(A+B) + C'(AB)`. These correspond to a logic **AND** for `C = 0` and a logic **OR** for `C = 1`.

- The paper also extends the DRAM structure to allow for logic **NOT** operations by connecting the inverted signal of the bitline to a dual-contact DRAM cell.

- The paper names the accelerator **Ambit** and shows that it can be easily integrated into a system, as its interface is identical to that of common DRAM, allowing for seamless integration.

- The paper shows that Ambit has both low hardware and control overhead compared to conventional DRAM.

- The paper evaluates the performance of Ambit by performing circuit-level SPICE simulations as well as analyzing real-world example problems simulated on the system. The SPICE simulation shows that the error rate is low for reasonable process variation and further elaborates that faulty modules could still be used as conventional DRAM, as the Ambit accelerator is an extension of conventional DRAM. The runtime performance analysis shows that for database access with bitmap indices and BitWeaving, Ambit provides significant speedup. Furthermore, a bit-vector implementation with Ambit is compared to an RB-Tree implementation, and it is shown that for large sets Ambit yields a significant speedup.

Strengths

Ambit uses existing technology by being an extension to conventional DRAM. It is also a very smart utilization of existing processes, resulting in a new way to compute within memory, allowing it to alleviate the memory bandwidth bottleneck for specific types of computations. The paper also proposes that faulty Ambit accelerators could still be used as conventional DRAM, thereby reducing yield costs for the manufacturer.

Weaknesses

The paper does not provide an actual manufactured Ambit accelerator, and therefore any of the results presented are based solely on simulations. It is fair to assume that these simulations are highly accurate, but simulated data can never fully replace real-world measurements; therefore, the claims must be taken with a grain of salt.

Ambit only implements a very limited instruction set with AND, OR, and NOT operations. While other binary operations can be constructed from these, it is also outlined that the successive loading of memory into the relevant DRAM lines is time-intensive. The paper itself does not propose any such implementation, and therefore I have to assume that the use case of the accelerator is limited to situations where these specific operations are the primary bottleneck.

My POV

I would see great value in testing a physical implementation of the Ambit accelerator to obtain real-world data to back up the simulations. In general, the paper alludes to steps that manufacturers still need to take to actually produce a working system, which somewhat undermines the simulated results.

Further exploration of chaining the AND, OR, and NOT operations would also have great value, as it would demonstrate to what degree more complex logic could be performed directly on the accelerator.

Takeaways

- Ambit introduces a very powerful solution to parts of the memory bandwidth bottleneck. The concept of computing not just *in memory* but *with memory* seems very powerful and worthy of further exploration.

- The results of the paper are based on simulation rather than physical integration, meaning that all results must be taken with a grain of salt.

Review 4: "A Logic-In-Memory Computer" by Harold S. Stone

Summary

This is a paper from 1970. It identifies memory bandwidth as a major bottleneck for computer systems and proposes the then-new idea of computing-in-memory. It also elaborates on possible implementations at a theoretical level.

- The paper explains that there are both magnetic and microelectronic types of memory, with the former being cheaper but slower and the latter being more expensive but faster. While this is not the same issue we face today — where the problem primarily arises from the distance between memory and CPU, as well as the technological differences between DRAM and SRAM — the general issue remains the same: we have a large, slow memory and a small, fast memory, and moving data between them is time-intensive.

- The paper proposes computing inside the cache, introducing a set of simple instructions that could be implemented. However, the paper is not a concrete implementation; rather, it proposes different use cases and solutions that could be achieved through in-cache computation.

- The paper acknowledges that it cannot make concrete predictions about the cost of in-memory computation and emphasizes that its feasibility heavily depends on the future development of the technology.

Strengths

The paper is almost prophetic in its prediction of the bottleneck caused by memory bandwidth. It also lays important theoretical groundwork for in-memory computing. Much of the groundwork done in this paper was later applied in other research, and therefore its impact can be felt even in modern state-of-the-art in-memory computation systems discussed in later papers.

Weaknesses

There are no clear weaknesses in this paper. The only limitation is that it was clearly ahead of its time, as the memory bottleneck would only become a major issue in the 1990s — a good two decades after the publication of this paper.

My POV

This paper had an incredible impact on the concept of in-memory computation. It makes very accurate predictions about future issues and proposes significant solutions, even without providing concrete implementations. The paper is an excellent example of proactive thinking — addressing an issue that may not yet be urgent but would become highly relevant in the future.

Takeaways

- This paper clearly demonstrates the merit of addressing and thinking about future problems before they become pressing issues.

- It very clearly establishes the foundations of in-memory computation as we know it today.

Review 5: "Processing in Memory: The Terasys Massively Parallel PIM Array"

Summary

The paper addresses the memory bottleneck. And proposes an in-memory-processing solution called Terasys. The paper provides a physical implementation as well as a C extension called dbC as an API with the accelerator. It also explores applications and gives a back of the envelope cost estimation. - The paper proposes Terasys an in-memory-processor that consists of a large number of processing elements (256x256) which are located in memory and perform single-instruction-multiple-data, abbreviated as SIMD, operations. All the processing elements are controlled by a global control unit. The system is scalable as the computational power grows in tandem with the memory. - The paper provides dbC which is an extension to C that allows interfacing with the Terasys accelerator. It introduces SIMD structures into C to allow tailored code to optimize the accelerator usage. - The paper outlines that Terasys has superior performance to conventional systems in image processing and matrix operations. As Terasys is located in memory the system achieves higher performance and lower energy usage than multiprocessor systems. - The paper provides a proof of concept real-world implementation to back up the claims in the paper. This backs up their claims about performance and power usage.

Strengths

The paper is an early instance of in-memory-processing. It provides a fast and working solution to real world problems such as the MRI imaging problem outlined in the paper. The paper lies important groundwork for later work in the area of in-memory-processing. Further it also provides a cost analysis giving transparency about use cases of the Terasys.

Weaknesses

The paper has fairly few concrete numbers and results. This is an issue as it complicates the assessment of the system's actual performance and viability in diverse scenarios. The proposed extension to C does not follow a general standard which can be more confusing then helpful in certain scenarios and which also greatly reduces the interoperability of the system. The paper also claims the system to be scalable and easy to integrate into greater systems. While their anecdotal evidence of integrating Terasys into an existing compute system and integrating it into a supercomputer seems to back this up, there is a lack of evidence to substantiate such a claim in its general form.

My POV

The paper was an early adopter of in-memory-processing and it is easy to propose improvements from a modern viewpoint. But this viewpoint wouldn't exist if not for papers such as this that lay groundwork for all the following research. While dbC did not catch on, the concept of in-memory-processing very much did. I would prefer it if the paper would have provided more actual data so that I could further assess the actual performance of the system. Nonetheless it is clear that Terasys's concepts were a success as we find similar architecture in modern state-of-the-art GPUs and compute capable DRAM.

Takeaways

- Terasys was an important early proof of concept for in-memory-processing that fuel research to this day and provided the foundation for many modern architectures.
- The paper highlights the importance of backing concepts with physical implementations to provide palpable results on which future research can build and to which it can compare itself.