1 Summary

Spending Moore's Dividend by James Larus (2009): Larus described that up to that point in time how decades of Moore's prediction, the improvements and doubling of transistors on chips every few years, had led to similar improvements in processor speeds, computer design, and microarchitectures; he called these Moore's Dividends. How Moore's Dividends were spent, he noted, were not previously quantified, but could be observed in the trends and behaviors of the software, namely, the increasing growth in resource requirements and computational capabilities expected of the computer, and the evolution of software practices. While these were not backed by evidence, they lined up with what is seen in Myhrvold's Laws and the cycle of computer industry innovation, which can be briefly summarized that software expands to fully (and perhaps overly) utilizes the resources offered by the hardware they run on, which in turn pushes the demand and pulls the market towards new and improved hardware, and and so on; essentially, for decades, both industries have in some way pushed the other to grow to meet Moore's Law. Larus used these observations to predict what impact they'd have on future software practices and parallel computing; his point being, that Moore's Dividends drove the direction and growth of software in the last few decades, but the technology of multicore processors and parallel computing will be what shapes future software practices.

2 Strengths

- The numbers and figures in the introduction and the examples/anecdotes in the body painted a very clear picture of what Larus meant by "how Moore's Dividend were spent", i.e. what aspects of technology changed and in what way as a result of the improvements seen by Moore's Law. Likewise for the "Future Implications" section.

3 Weaknesses

- The article was long-winded in building up to the final point, in particular, I don't think it needed as much data as it had to set up the point in the introduction that Moore's Law was observed in real computers over several decades.
- Some of the figures included were stylized from their original source, but in a way that was confusing to read without hunting for the context in the article (missing axis labels and confusing captions).

4 Rating: 3

5 Comments

A lot of the topics Larus brings up are familiar as to what a student would have been exposed to so far throughout their education and pieces them together nicely, which I think is important for building context with the more advance material we go on to learn; for example, when he described how software practices were shaped (or enabled) by the improved processor technology, many ring true to what is seen in industry and schools today (object-oriented design, modularity, abstraction, richer libraries, emphasis on security, reliability, compatibility, prevalence of higher level languages, compilers and performance tuning, etc.). I believe he was correct in his own observation that certain trajectories of software development wouldn't change very much and continue to grow in their resourece demands as software continues to become more complex. While the article sets a nice stage of the state of technology at the time with plenty of examples, it is also over 10 years old. More can now be expanded on his observation on how multicore processors and parallel computing would be the next "revolution" to change software in the way integrated circuits and transistors had in the past. In particular, the other article "We're not prepared for the end of Moore's Law" (David Rotman, 2020) linked for this week's reading made a few similar points to Larus, such as how software still does not do enough to fully utilize the architeture of multicore and parallelism because programmers don't focus on efficient code in favor or relying on the compilers optimizing and processors getting more powerful. More than that, other means of innovation besides parallel programming have garnered more interest since Larus' article. One that Rotman notes is the growth of specialized architecture and advance software that works with them, which was mentioned in the 2017 Turing Lecture (Hennessy and Patterson) and that Larus does not touch on, nor did Larus mention the idea of the "decline of computers as a general purpose technology". That's not to say parallel programming isn't increasingly important because multicore processors are the norm; but rather, its not certain what the next "successor technology" or what the future landscape of computing would look like if Moore's Law trends to a halt. Parallel programming also may not have reached the prevalence Larus had hoped it would, but $\sim$15 years may not have been enough time to make that determination. However, I think the 2009 article is still less relevant 10 years after it was published and there are more current ones that could cover what Larus meant to cover and to include more recent developments in technology trends.

References

[1] James Larus. 2009. Spending Moore's dividend. Commun. ACM 52, 5 (May 2009), 62-69. https://doi.org/10.1145/1506409.1506425

[2] David Rotman. We're not prepared for the end of Moore's Law. MIT Technology Review. 2020.