HARDWARE BASED ACCELERATION OF APPARENT NETWORK SPEED USING THE SEQUENTIAL CACHING TECHNIQUE

Chandrasekhar Gudipati
Student,
Department of Electronics and Communication Engineering,
Manipal Institute of Technology, Manipal

Dr Rajesh Challa
Architect
Samsung Research Institute Bangalore

H Srikanth Kamath
Assistant Professor-Selection Grade,
Department of Electronics and Communication Engineering,
Manipal Institute of Technology, Manipal

Abstract - 5G today has become synonymous to technological and economical improvement. It is therefore of utmost priority to perfect the system before it reaches ubiquity. This paper aims to provide a cost-effective hardware solution for cell towers that allows for a higher end- user network speed without making any ultra-expensive improvements to the server hardware. All that is required is a simple low capacity and inexpensive storage system added to each cell tower. Multiple simulations have been performed using Python to prove that higher speeds can be achieved using a technique that is termed in this paper as 'Sequential Caching'. The reason behind this nomenclature will become more apparent in the later segments of this paper.

Keywords- Sequential Caching, Resource as a Service (RaaS), Edge Computing, Fog Computing

Classification- Wireless Communication Technologies

1. INTRODUCTION

5G is almost always associated with the next wave of smart technology. There are many significant advances being made in 5G technologies [1][2]. 5G is different from previous generations of wireless communication in that it requires a lot more infrastructure (and hence capital expenditure) to cover the same area as a conventional 4G or 3G network. This however provides us with a unique opportunity to focus on hardware acceleration at each individual tower level simply because there are so many of them and so even a small performance increase at the tower level would boost the system metrics by a large amount. 5G towers cost a lot more than their 4G counterparts and hence technological add-ons to these antennas are of little expense since they

would not mean much in relation with the price of the antenna itself.

1.1. ULTRA-DENSE NETWORKS IN 5G

5G networks are expected to be much denser than their 4G counterparts [3] as stated earlier in brief. This already adds to the stress on servers. Too many towers await responses from the same server. Atop of this, 5G is poised to have more and more end devices which just adds to the network stress for the servers. With all these problems to be seen frequently, it becomes of paramount importance to find a way to distribute this traffic so that the servers can function optimally.

1.2. NETWORK SPEEDS IN 5G

5G network speeds in the near future will be in excess of 1Gbps [4] and perhaps significantly higher [5]. This high speed means that more data can be transmitted per unit time. So, in a round robin fashion with multiple network towers, a lot of data can quickly be transmitted to each tower during its turn in each round. Most major network towers have wired connections. Even today, the general standard 10 Gigabit Ethernet (IEEE 802.3ae) offers speeds of up to 10Gbps. By the time 5G reaches ubiquity, this speed is bound to increase far beyond the maximum limits of 5G. How we utilize this excess speed becomes of paramount importance.

1.3. OBJECTIVE OF THE PAPER

The objective of this paper is to explore the effect that adding a small memory module to each network tower will have on the network speed experienced by end user devices. By conducting simulations on various memory sizes, an ideal memory size can be estimated based on cost to performance ratios.

2. BACKGROUND THEORY

The server to tower connections are always faster than the tower to user connections. This means that the tower to user connections act as continuous bottlenecks to an otherwise fast network. The entire objective of this research is to allow the entire network (including the tower to user communications) to be as fast as the server to tower communications themselves.

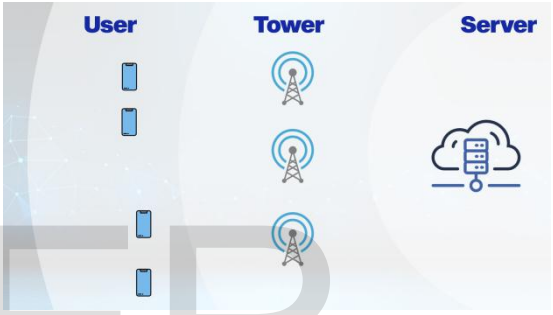

Fig 1: General Network Structure

While software methods might help[6], one can do only so much without hardware changes. The problem with hardware upgrades is that they are generally prohibitively expensive. Sequential caching is a rare example of a relatively cheap hardware enhancement.

2.1. THE PROBLEM WITH CONVENTIONAL TECHNOLOGIES

Almost every major application and company uses its own in house servers. Servers are often overburdened with the number of requests coming at their direction. This means an overall delay for the user which might cause the user to quit the service. In other cases, the entire application might be rendered useless due to the instantaneous requirements that it demands. For example, live multiplayer

gaming in applications such as Google Stadia. If in view of this, a company invests in multiple servers beyond the general traffic, it is a waste of valuable resources, not to mention an unnecessary expense.

2.2. IDEAL SOLUTION

To avoid the above mentioned problems, it is important to have a unified resource that either helps to smooth out the high spikes of instantaneous traffic by providing memory access to the towers or one that provides a simple open access computing platform that performs at least basic parsing and preprocessing of data. Put rather simply, the more locally we perform these operations [7][8], the lesser the burden will be on the platform.

2.3. EDGE COMPUTING

Edge computing [9] is often seen as a quick solution which is practically free for network providers as well as service providers. However, this cost is usually borne by the end user, or in this case, the general public. Most people do not require the high speeds that are demanded by professionals who use 5G for advanced applications. Also, not all programs need to make use of extreme 5G speeds. For example, email services can have a delay of a few seconds without causing any issues to anyone.

It is unfair to charge these expenses onto the end user who may or may not avail these services. Every type of business gains from shifting to extreme edge computing. Device manufacturers can make more money due to the increased computational requirements of phones and laptops for which they will charge the end user a premium. Service providers will save a lot of money since they will not need to invest too much money into computational servers. Finally, network providers can say that the same application runs quicker without actually investing any money on resources. So, while all types of businesses will definitely make profits from increasing the focus on edge computing, it is very much against the interests of the final customer to increase the costs of the end user devices. Hence, a different approach is necessary to solve the same problem with a similar kind of results.

2.4. SEQUENTIAL CACHING

The idea of sequential caching is to enable resources similar to fog computing. This is done by allowing the towers to form an intermediate or fog layer that allows for semi local computing. Memory modules are attached to the hardware of cell towers. In a round robin scenario, each cell tower gets a particular amount of time before the server moves on to cater to the needs of the next tower. Finally, once it caters to the need of every other server, it returns to the first tower. Now, in a conventional scenario, the amount of information that is transmitted from the server to the tower and then to the end device (say a mobile phone or a laptop) is limited due to the relatively lower speed of Tower to End Device speed when compared to the Server to Tower speed. To overcome this problem, Sequential Caching uses memory modules at each cell tower as shown in the figure.

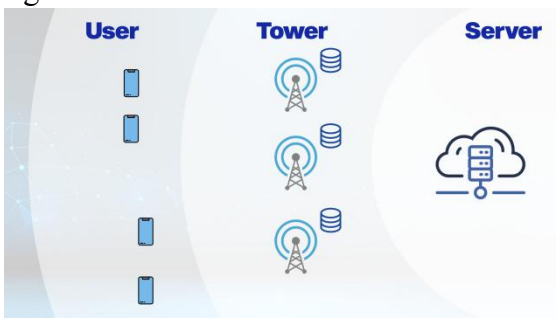

Fig 2: Basic Assumed Network Structure

Let us call the uppermost tower in the figure as Tower 1, the middle tower as

Tower 2 and the bottom tower as Tower 3. In a conventional round robin fashion, the server first handles the requests of Tower 1 up to a duration of time, say 't'. Since Tower 2 has no devices and hence no requests, the Server skips it and moves onto Tower 3. It serves Tower 3 for a limited amount of time before moving back to tower 1. It is to be noted here that the entire process moves at the speed of the tower to end device connection. For example, is the Server to Tower Connection is at a speed of 200Mbps and the Tower to End Device Connection is at a speed of 150Mbps, conventional methods loose out on that 50Mbps speed every time the server responds to the requests of the tower during its turn in the round.

On the other hand, when sequential caching is used, the tower collects information at the entire possible Serwer to Tower Speed and whatever cannot be transmitted at any instance will be stored in the buffer memory. The buffer memory is like a queue data structure, which allows data to be sent to the tower sequentially. This continues until the buffer memory is filled. At this point, the speeds reduce down to conventional levels. The good news is that since each turn in the round robin method is only for a very short duration of time, for most of this time, the buffer memory will not be full, especially when the memory size is in the order of gigabytes. And even if the buffer memory is full, the worst that will happen is that the speeds reduce to conventional levels only till the end of the turn. This is because the same size of data entering the buffer will be leaving the buffer.

It is best to understand this concept with the same example mentioned for the conventional method. Each data packet that reaches the tower enters this buffer memory at the tail end with the full 200Mbps speed. Out of this, a portion is transmitted at 150Mbps speed from the head end while the remaining information is stored in queue in the buffer. Meanwhile, the next packet of information arrives at the tail end from the server at 200Mbps while the head end continues to send out data to the user device at 150Mbps. This continues until either the buffer is full or the tower's turn ends. If the buffer is full, the tower will inform the server to reduce the speeds to the conventional 150Mbps until the end of its turn. If the tower's turn ends and the server moves onto the next tower, the tower will send out the data remaining in the buffer to the user device at the conventional 150Mbps. This is the key part about sequential caching. The time that the device spends waiting again for its turn is utilized in this time gap in sequential caching, which in turn increases the apparent speed at the user end.

3. RESEARCH METHODOLOGY

To understand the simulations, it is important to go through the Sequential Caching network structure envisioned in the theory section of this paper.

Python was used for the simulation within the Jupyter Notebook environment. For the simulation, since future values of network speeds cannot be accurately assumed, general network speeds were assigned, but always keeping the Server to Tower speeds faster than the Tower to User speeds. A little random error was also added to the datapoints. This was done to simulate a real world scenario. To ensure this, random functions were enabled within specific limits, ensuring that the lower limit of the Server-Tower boundary was more than the higher limit of the

Tower-User speeds. Figure 2 is a highly simplified example of such an assumption.

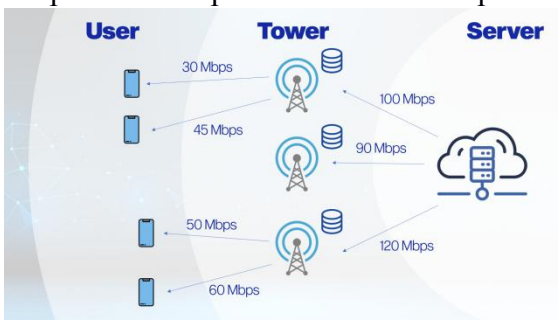

Fig 3: A Visual Example of General Speeds

Initially, the simulations were performed with a fixed memory size, 1GB to be more precise. Afterwards, different data sizes were explored to find which memory size was ideal for the scenario. The sole objective of this experiment was to prove that the model actually works, and real-life data in a particular scenario collected from surveys can be incorporated for accurate results for that particular scenario. Then, the ideal memory size for that particular scenario can be calculated.

4. RESULTS

The results obtained were very convincing. To start things off, the simulation was performed with an infinite buffer memory size. This provided around a 400% increase in apparent speed at the user end.

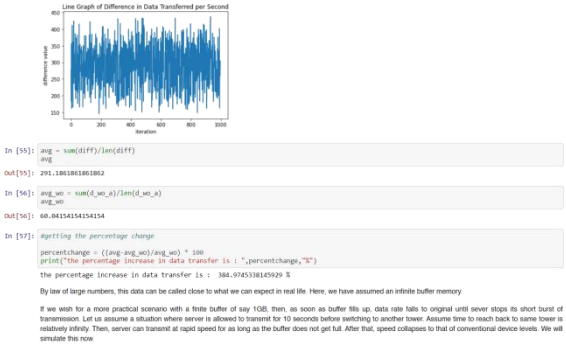

Fig 4: Simulation with Infinite Memory

This however is not a practical solution and so the simulation was performed again, but this time, 1GB of buffer memory was arbitrarily assigned. This 1GB of memory provided about a 50% gain in apparent network speed in my simulations.

```
In [59]: avg_q = sum(q)/len(q)
         avg_q
Out[59]: 50.110386491279925
```

Fig 5: Simulation with 1 GB (~50% increase)

These two simulations prove that the entire system works. The next objective was to test for the best memory value for optimal results. For this, three new scenarios were analyzed after the original ones. In the third scenario, the buffer memory size was varied from 0GB to 10GB, increasing by 1GB at a time. In the figure, the X axis represents the memory size and the y axis represents the percentage increase in performance. In mathematical terms, this would mean

$$pis = \frac{s_s - s_o}{s_o}$$

Equation 1: Percentage Increase

Where
pis: Percentage Increase in Speed
s_s: Speed with Sequential Caching
S_0: Speed without Sequential Caching

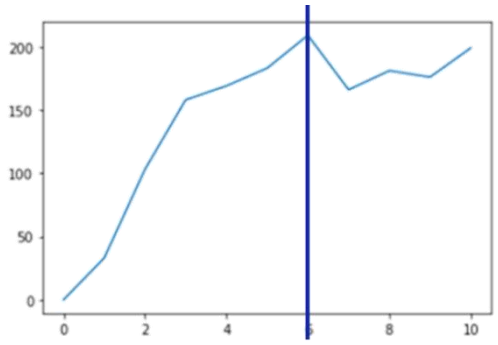

Fig 6: Results of Third Simulation

In the fourth simulation, the upper memory limit was increased from 10GB to 20GB. The results were consistent with the second simulation. After a point, the average speed was just oscillating.

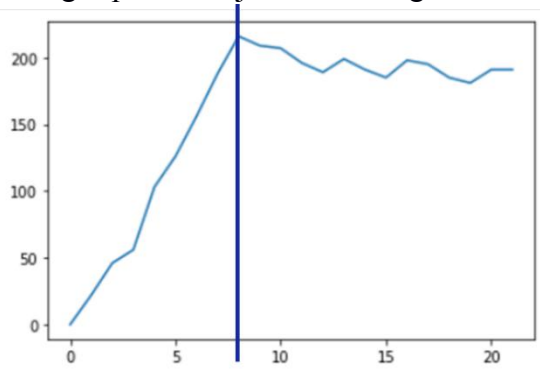

Fig 7: Results of Fourth Simulation

In the fifth simulation, the granularity of the simulation is increased. Instead of a 1GB increase in data, I chose a 0.5GB increase in data. This way, a more accurate optimal value can be ascertained.

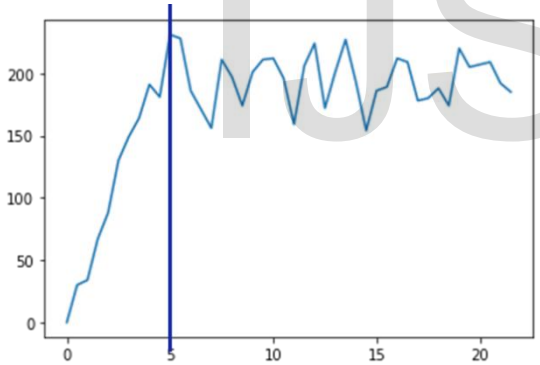

Fig 8: Results of Fifth Simulation

Now that we have covered the results of the code, it is time to figure out why this oscillation in performance is happening beyond the optimal value.

4.1. RESULT ANALYSIS

It is safe to say that these results were perfectly anticipated beforehand. The reason why there is no significant improvement after the optimal point is that once the buffers are full in a tower, speeds are expected to drop back to conventional levels. So irrespective of how large the buffer gets (within a reasonable range), there will not be any change in data speeds. Since our simulations assumed a one-second round robin system, where each tower gets one second of time before the server moves onto the next tower, the optimal point was enough buffer to store the entire information for this one second. After that point, the entire buffer is not used since the server moves onto the next tower. This simply amounts to resource wastage. In essence, the shorter the time allocated to a single tower in the round robin fashion, the smaller the size of the buffer necessary and hence the cheaper the network construction gets.

5. IMPROVEMENTS TO THE MODEL AND FUTURE WORK

While the entire memory buffer model works really well, there is no reason to halt progress at this level. There are two main areas where significant improvements can be achieved, both performance and price wise.

5.1. PROCESSOR BASED SEQUENTIAL CACHING

A processor can be used instead of a memory system. This way, a lot of preprocessing of data can happen to the data packets within the tower itself while in wait for the server to pick them up. Only the computationally intensive parts of the processing will be done in the server. This is a great option for gaming and other interactive screen rendering technologies, where the objective is to minimize the load on the server.

For example, if a user is playing a server based game, the server automatically expects a few next moves and sends perhaps a repetitive preprocessing code that can be executed in the tower itself.

On the other hand, the tower processor can be used simply for light post processing and rendering purposes. Either way, it decreases the load on the server and the time taken for the code to wait for the processing (this time is used by the tower preprocessor).

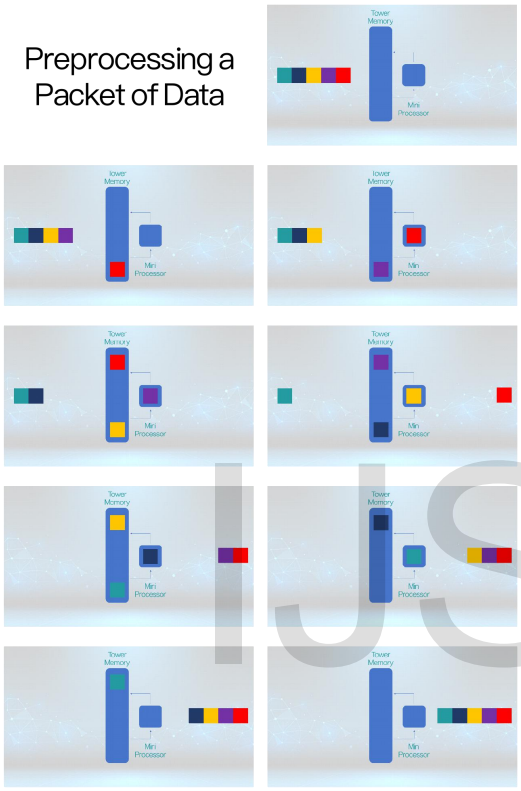

Fig 9: Preprocessing at the tower.

Each packet enters the memory buffer. One after the other, the packets enter the mini processor. The tower then requests the server for commands regarding preprocessing. Once received, the operations are performed on the packet after which the packet heads back to the tower memory and the next packet arrives at the mini processor. It is to be noted that it is simpler for the server to send commands to the tower than to actually execute them in house. This way, the burden on the server resources is significantly decreased and it can handle a lot more inputs than it normally can using the existing conventional techniques.

It would be better for the processor to be a CPU instead of a GPU since CPUs perform better when it comes to a wide variety of functions instead of one specialized parallelized function, which is rarely the case in preprocessing.

5.2. USAGE BASED ALLOCATION OF MEMORY

It is not necessary for every tower to have the same memory size. Instead, we can have a system where cities and urban conglomerations have the highest optimal memory (around 6GB in our case) and allow rural regions and places with fewer mobile data users to have a smaller memory size (around 1-2GB) or to avoid the memory altogether. These regions can continue with conventional techniques.

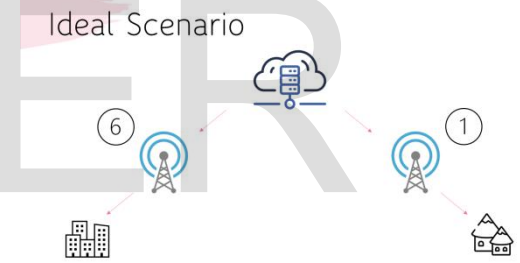

Fig 10: Usage Based Allocation of Memory

5.3. MONETARY POLICY

No model is viable without a healthy monetary policy that benefits all stakeholders. Sequential Caching may be used as a Resource as a Service (RaaS) offering by the network provider. Just how data charges apply to the customer, memory usage charges must apply to servers.

Fig 11: Monetary Policy

This way, enterprises can decide for themselves whether they want to use the memory RaaS or not. The customer will in no way be financially impacted by this decision. The competitiveness of companies will be decided by their decision to embrace this faster model and this will move the entire market towards this model.

5.4. USER PRIVACY AND DATA SECURITY

User privacy and data security are very important aspects that need to be addressed when a third party such as the network provider stores a lot of data, even if momentarily. Since most services and applications offer an encryption system for user data anyway, this is not a big issue when dealing only with memory buffers. Even today, cell towers handle data which is encrypted by applications or services and not the network providers.

The issue becomes important when we add processing capabilities to the towers. Here, in order to comply with cybersecurity practices, a black box system must be created so that no outsider can enter the hardware and steal the data being processed. The system that is being proposed here is similar to that of Google Colab or IBM Watson, where user code and data are processed without any threat to data security or loss of privacy.

6. CONCLUSION

Sequential caching is an inexpensive solution to the problem of bottlenecks in networks, whether 4G or 5G. 5G network components are still being built and by ensuring early adoption of Sequential Caching, there is a good chance that this might become a viable technique to drastically increase network speeds. It is also a rare combination where all stakeholders benefit both monetarily and performance wise. Future enhancements and fine tuning of sequential caching might just be the key to a great 5G experience. Sequential caching is a general purpose technique meaning that it is very simple to integrate it with more specialized equipment that improve very particular performance metrics. Other network hardware can be used alongside sequential cache memory without any impedance to its performance. Using software enhancements in tandem with the Sequential Caching Technique will lead to the best results.

7. REFERENCES

[1] P. Pirinen, "A brief overview of 5G research activities," 1st International Conference on 5G for Ubiquitous Connectivity, 2014, pp. 17-22, doi: 10.4108/icst.5gu.2014.258061.

[2] Nam Tuan Le, Mohammad Arif Hossain, Amirul Islam, Do-yun Kim, Young-June Choi, Yeong Min Jang, "Survey of Promising Technologies for 5G Networks", Mobile Information Systems, vol. 2016, Article ID 2676589, 25 pages, 2016. https://doi.org/10.1155/2016/2676589

[3] Jialing Liu, Huawei US Wireless Research and Standards, Weimin Xiao, Huawei US Wireless Research and Standards, Chih-Lin I, China Mobile Research Institute, Chenyang Yang, Beijing University of Aeronautics and Astronautics, Anthony Soong, Huawei US Wireless Research and Standards

[4] Gopal, B G. (2015). A Comparative Study on 4G and 5G Technology for Wireless Applications. IOSR Journal of Electronics and Communication Engineering (IOSR-JECE). 10.

[5] Yuki Arikawa, Hiroyuki Uzawa, Takeshi Sakamoto, Satoshi Shigematsu, Shunji Kimura, High-speed radio-resource scheduler with hardware accelerator for fifth generation mobile communications systems, IEICE Communications Express, 2017, Volume 6, Issue 5, Pages 236-241, Released May 01, 2017

[6] B. Ma, W. Guo and J. Zhang, "A Survey of Online Data-Driven Proactive 5G Network Optimisation Using Machine Learning," in IEEE Access, vol. 8, pp. 35606-35637, 2020, doi: 10.1109/ACCESS.2020.2975004.

[7] U. Niesen, D. Shah and G. W. Wornell, "Caching in Wireless Networks," in IEEE Transactions on Information Theory, vol. 58, no. 10, pp. 6524-6540, Oct. 2012, doi: 10.1109/TIT.2012.2205733.

[8] Ma, Ge & Wang, Zhi & Ye, Jiahui & Zhu, Wenwu. (2018). Wireless Caching in Large-Scale Edge Access Points: A Local Distributed Approach. 726-728. 10.1145/3241539.3267741.

[9] Wang, Zhi & Zhu, Wenwu & Sun, Lifeng & Hu, Han & Ma, Ge & Ma, Ming & Pang, Haitian & Ye, Jiahui & Li, Hongshan. (2021). Multimedia Edge Computing.