System Considerations for ngVLA Data Processing, Transport and Storage Systems
ngVLA Computing Memo 12

Rafael Hiriart

September 4, 2024

1 Introduction

We develop in this memo a parametric model of the ngVLA data processing, transport and storage systems, with the intention of identifying and quantifying (where possible) several system design tradeoffs and areas of technical risk. We define this model as implementation-agnostic as possible, to avoid unnecessarily constraining the design space being evaluated by DMS, who is responsible for the design of this part of the system. We provide several quantitative examples to develop an understanding of the orders of magnitude involved on these system parameters, and as a sanity check for the parametric model. These examples shouldn't be considered as estimations to be used for budgetting purposes. They depend on design choices, implementation details, and market trends that are not in the scope of this memo. The detailed design, along with a carefully developed budget for the transport, processing and archiving systems is DMS responsibility. The goal of this parametric model is to identify technical risks, helping to guide the prioritization of design activities for this part of the system.

2 System Model

We model the system as shown in Figure 1, adopting the notation and a few results from basic queue theory to characterize some important system parameters, such as the size of the storage buffers and the number of processors required to process the observed data. The basic terminology is shown in the upper left corner. A queue can be characterized by an input rate λ, service time S and throughput $X = 1/S$. The total number of requests in the queue, including the one being serviced in the server is Q, while the total time or *latency* that it takes for a single request to pass through the entire queue is R. This is a stochastic model, so all these parametes are random variables.

The final system will probably be more complicated than shown here, once all design requirements are considered. Usually storage systems adopt a hierarchical design. It would be very expensive to provide a high-bandwidth and high-capacity system at the same time, so it is common for high-performance storage systems to integrate a fast but relatively low-capacity "scratch" area, backed by high-capacity but low-bandwidth "work" area. Similarly, very big archives commonly use a relatively cheap but slow tape system for massive storage (a must in our case due to the electrical power operational expense that a disk-based system would impose), which works with a disk-based front buffer to read files into and

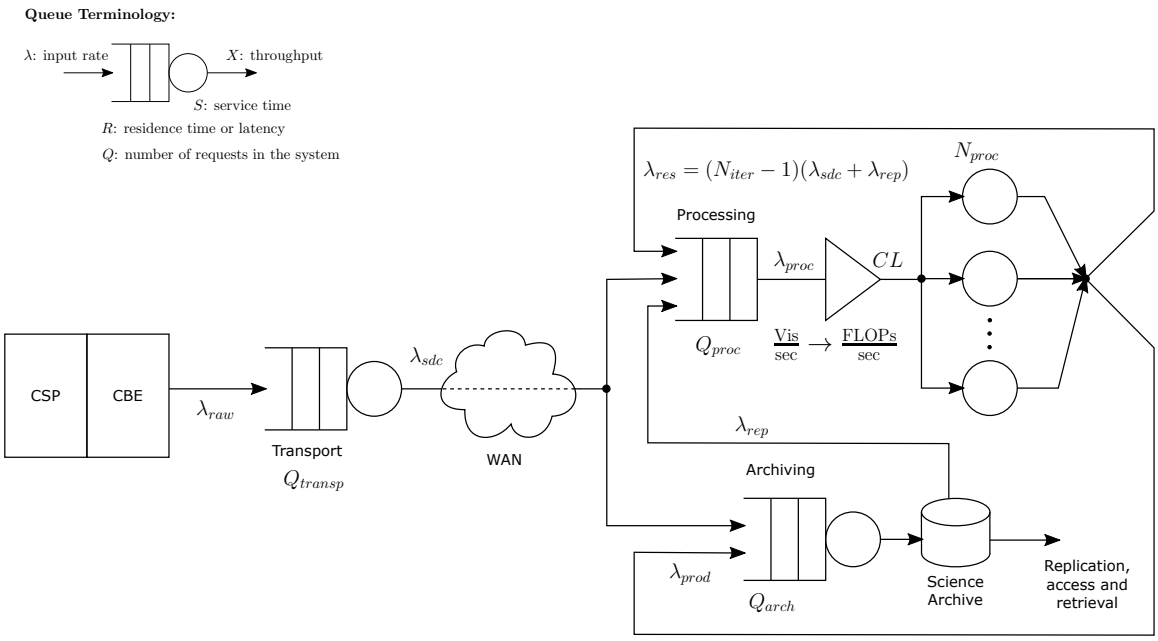

Figure 1: A simple system model.

write files out of the tapes. We don't consider all these details, which will need to be designed/optimized by technical experts in DMS.

The CSP will generate raw interferometric data that will be received by the CSP Back-end computer cluster, which formats the data and performs the last steps of processing (e.g. averaging, frequency stitching, and RFI flagging) before dumping the raw-data datasets into the "Transport" buffer, from where it will be transmitted to the Science Data Center (SDC) across a Wide Area Network (WAN). The data rate flowing into this storage buffer is λ_{raw}, and the data rate flowing out of the storage buffer is λ_{sdc}. This is the input data rate into the SDC, and it will depend on the network connectivity available from the Central Electronics Building to the SDC.

The data flowing into the SDC is both processed and archived. The exact configuration of how this happen is not important for the purposes of calculating the parameters of interest. We could archive the data first and then process, or archive and process the data in parallel, or some hybrid combination. Either way both the processing system and the archiving system receive the same input data flow λ_{sdc}. We measure this data flow either in GVis/sec or GBytes/sec.

Before entering the processor pool, we perform a "unit" conversion (represented by the amp in Figure 1), transforming the input data rates in GVis/sec to FLOPs/sec, as this is the proper unit to characterize the processing server throughput. As described in [2], this is not just a multiplication by a constant, like the transformation from GVis/sec to GBytes/sec, but the conversion will depend on the algorithmic requirements of the datasets to process (in other words, the gain of the amp is variable). The data rate λ_{proc} is converted into the computing load CL. This computing load is serviced by a number of processors N_{proc}. At this point we just talk of these servers as *processors*. Their exact nature is a design detail left to be decided by DMS experts. We conform ourselves by pointing out that they will probably be composed by GPUs, given the current trends in \$/FLOP and Watt/FLOP, and the fast pace of improvement of these systems, due to the intense competition fueled

by recent AI developments. Likewise, the processing buffer could be constituted by a large parallel filesystem, or distributed as local storage attached to the processing nodes, or a combination of both.

Given the iterative nature of the imaging algorithms, the same visibilities are processed several times. We represent this as a feedback loop from the output of this multi-queue system back to its input, with a data rate λ_{res} (for "resident"). We can think that the input data rate is partitioned in "chunks" of visibilities that go through the major cycle calculations several times before exiting. What exactly is a "chunk" is another detail left to be defined by the DMS experts. It will probably depend on how the parallelization axes (time/scans, frequency, polarization, etc.) should be partitioned to optimize the utilization of the server cluster, taking into account their memory restrictions, I/O bandwidth, etc. A multi-processor server can be modeled simply as a single processor system with a service time $S' = S/N_{proc}$.

We also consider as input to the processing system the re-processing data rate λ_{rep} (in Vis/sec) coming from the Science Archive. This has been nominally set to 20% in the system requirements. The output of the processing system is a stream of high-level data products that is sent to the Science Archive, which from [2], should be less than 10% of λ_{sdc}.

All the queues in Figure 1 need to comply with the stability condition

$$\lambda \leq X \tag{1}$$

in the average, otherwise their length becames unbounded. When a queue is stable, the average output data rate is equal to the average input data rate.

A general relation, valid for any probability distribution on the input data rate and service time is Little's Law:

$$Q = \lambda \cdot R \tag{2}$$

that is, the average number of requests in the queue is equal to the product of the expected value of the input data rate and the expected value of the latency.

The fraction of time that a server is busy is the server utilization, defined as $\rho = \lambda/X$. For a stable queue, $0 \leq \rho < 1$.

3 The transport system

For the transport system, the throughput is determined by how much network bandwidth it will be procured between the Central Electronics Building and the Science Data Center. The location of the Science Data Center hasn't been defined yet, so we cannot provide a firm estimation of this cost. Notionally we assume the cost of a 100 Gbps WAN connection to be around $5,700 per month, based on [5].

It is possible to show[1, 3] that the average number of requests (bytes) waiting in a queue to be serviced (transported) is

$$Q = \frac{\rho}{1-\rho}. \tag{3}$$

This allow us to calculate the average utilization of the transport buffer. However, this is the *average* buffer utilization, not the maximum buffer utilization when the fluctuations of the input data rate — determined by the differences in the ROP/EOP use cases — are

taken into account. The transport buffer needs to be sized to the maximum queue length plus some margin, as we cannot afford to lose science data due to the buffer overflows.

One option would be to procure network connectivity to match the peak data rate, which from [2] is 323.6 GBytes/sec for the EOP. This would be quite expensive, at $1.7MM per year, given the characteristics of the input data rate distribution. Note that most of the time this large networking bandwidth would sit unused, so this is clearly not an cost-efficient solution. On the other hand, if we size the transport bandwidth for less than the peak data rate, then the transport buffer needs to be sized so it will never overflow, and it becomes necessary to define for how much time the system needs to stand the peak data rate in this condition of unstability. This parameter will determine the cost of the storage buffer, which can be quite substantial if we need to stand that the observations that generate extreme data rates are observed close to each other.

An important characteristic of the system is that we can control the input data rate through observation scheduling. In other words, the scheduling system can estimate the data rate that the candidate SBs will produce and knowing the status of the storage buffer, schedule only observations that will not overflow the system, allowing the system to recover its capacity by observing relativey low data rate observations for some time.

If a system observes at λ_{peak} for a time T_{peak}, with a queue being initially at the average value Q_{avg}, then it will reach a maximum queue size of

$$Q_{max} = Q_{avg} + (\lambda_{peak} - X_{sdc}) \cdot T_{peak} \tag{4}$$

where X_{sdc} is the WAN bandwidth to the Science Data Center. Then if, after reaching the value Q_{max}, the system observes low rate cases which generate a rate λ_{low}, then the recovey time is

$$T_{rec} = \frac{Q_{max} - Q_{avg}}{X_{sdc} - \lambda_{low}}. \tag{5}$$

We define C_{stor} as the cost of storage in $/GByte, and C_{net} as the cost of WAN networking in $/GByte/sec. As the networking connectivity is a service that it is payed monthly (or annually), not a construction cost, for the purpose of comparing costs we define C_{net} as the Present Value (PV) of all these monthly payments for the lifelong of the project.

With this, the cost of this part of the system is

$$\begin{aligned} C &= C_{net} \cdot X_{sdc} + C_{stor} \cdot Q_{max} & (6)\\ &= C_{net} \cdot X_{sdc} + C_{stor} \cdot \left(\frac{\lambda_{avg}}{X_{sdc} - \lambda_{avg}} + (\lambda_{peak} - X_{sdc}) \cdot T_{peak} \right) & (7) \end{aligned}$$

where we have used equation 3 to replace Q_{avg} noticing that $\rho = \lambda_{avg}/X_{sdc}$. Rearranging this equation:

$$C = (C_{net} - C_{stor} \cdot T_{peak}) X_{sdc} + C_{stor} \left(\frac{\lambda_{avg}}{X_{sdc} - \lambda_{avg}} + \lambda_{peak} \right) \tag{8}$$

Deriving this equation with respect to X_{sdc} and equating it to zero:

$$\frac{d}{dX_{sdc}} C = C_{net} - C_{stor} \cdot T_{peak} - C_{stor} \cdot \frac{\lambda_{avg}}{(X_{sdc} - \lambda_{avg})^2} = 0 \tag{9}$$

we find an optimum for the network bandwidth that should be procured:

$$X_{sdc} = \lambda_{avg} + \sqrt{\frac{C_{stor} \cdot \lambda_{avg}}{C_{net} - C_{stor} \cdot T_{peak}}} \tag{10}$$

Note that if $C_{net} - C_{stor} \cdot T_{peak} > 0$ then we can find an optimal value for X_{sdc}. If not, it is apparent from equation 8 that the cost always decreases when X_{sdc} increases, so in this case the optimal solution is for the network bandwidth to be as large as λ_{peak}, with $Q_{max} = 0$.

Let's see an example. Assume that the cost of networking is $5,700 per month per 100 Gbps, or an annual cost of $68,400. With a return rate of 6%, this gives a PV for 20 years of $852,942 for 100 Gbps, and $C_{net} = 68,235$ $/GBytes/sec. Let's assume $C_{stor} = 0.4$ $/GByte, $\lambda_{avg} = 21.0$ GBytes/sec (which is the case for the EOP), and $T_{peak} = 4$ hours. This produces an optimal value $X_{sdc} = 21.01$ GBytes/sec and $Q_{max} = 4.36$ PBytes with a cost of $1.74MM. The cost of networking is $1.4MM, and the total cost is $3.2MM.

In this example, the networking bandwith is very close to the average input data rate, because of the cost structure expressed in the second term in the right hand side of equation 10. This function behaves almost like a step function: as long as $C_{stor} < C_{net}/T_{peak}$ the optimum bandwidth is close to λ_{avg}. When $C_{stor} \geq C_{net}/T_{peak}$, the bandwidth should be equal to λ_{peak} and no storage buffer is necessary. If we neglect Q_{avg} from equation 4, as this term is relatively small, then the optimum X_{sdc} becomes a step function.

In practice, the size of the transport buffer will also need to be sized to stand outages on the WAN connectivity service. If we assume a Mean Time To Repair (MTTR) of 3 days, then the transport buffer need to be at least $3 \times 24 \times 3600 \times 21.0 = 5.4 \times 10^6$ GBytes, or 5.4 PBytes, to allow observing to continue. This is in the same order of magnitude that the value calculated before. We should use the maximum of these two values, although a trade that should be evaluated is the possibility of contracting redundant network links instead. Ideally these redundant links should be independent and follow different paths from each other. An hybrid solution will probably be necessary, taking into account that the MTTR could be higher than 3 days.

4 The processing system

We can calculate the value of λ_{res} using Little's law. It is easier to visualize how the processing system works if we think that a number of processors will be assigned to process a "chunk" of visibilities, which will generate a number of FLOPs that need to be computed. These chunks parallelize the computation of a major cycle. Little's law estates that the number of requests in the system Q is equal to the input data rate λ times the latency R, that is:

$$Q = \lambda \cdot R \tag{11}$$

Let's say that after assigning a number of processors per chunk, the latency to process each chunk is R_{chunk}, for each major cycle. The total latency of each chunk will be R_{chunk} times the number of iterations N_{iter} that the same visibilities circulate inside the system. So, the total number of chunks being processed inside the system at any given time is

$$Q_{chunk} = \lambda_{chunk} \cdot N_{iter} \cdot R_{chunk} \tag{12}$$

where λ_{chunk} would be the the sum of λ_{sdc} and λ_{rep}. This can be reorganized as

$$Q_{chunk} = (\lambda_{chunk} + (N_{iter} - 1) \cdot \lambda_{chunk}) \cdot R_{chunk}, \tag{13}$$

to obtain the resident data flow:

$$\lambda_{res} = (N_{iter} - 1) \cdot \lambda_{chunk} = (N_{iter} - 1) \cdot (\lambda_{sdc} + \lambda_{rep}). \tag{14}$$

If the number of iterations is 1, then this flow is zero.

An important point to highlight is that the number of processors assigned to process each chunk is not really significative for the total throughput, although it affects the latency of processing each dataset. In fact, the total number of processors being used is the number of chunks multiplied by the number of processors assigned to each chunk. If we decrease the number of processors assigned to each chunk, the latency increases proportionally and the number of concurrent chunks in the system increases proportionally as well, cancelling any effect in the throughput. The total number of processors required stays the same.

However, this assumes a proportional reduction in the latency with the number of processors, or in other words, a linear speedup. As we have seen before, the speedup doesn't necessarily improve linearly with the number of processors. This is an important factor to consider in deciding the number of processors assigned to each chunk. In order to reduce the latency, we want to parallelize as much as possible, but the degree of parallization will be bounded by the saturation of the speedup. This fact may impose bounds in the achievable latency.

The total aggregated bandwidth required by the N_{proc} processors at any given moment is

$$BW \left[\frac{\text{GBytes}}{\text{sec}}\right] = \frac{\lambda \left[\frac{\text{PFLOPs}}{\text{sec}}\right] \cdot 10^6}{AI \left[\frac{\text{FLOPs}}{\text{Byte}}\right]} \tag{15}$$

For each floating point operation that the system needs to compute, a number equal to the arithmetic intensity AI bytes need to be loaded. This is the aggregated bandwidth measured at the interface between the processor registers and the rest of the memory/storage hierarchy (L1-L3 caches, main memory, disk storage). Because of the nature of the algorithms (represented by factors K_e and K_c in [2]), the same visibility numbers are retrieved multiple times to participate in the gridding operations, with the outcome that the aggregated bandwidth at the processors ends up being much higher than the output bandwidth from the CSP. For example, even with the system operating at a modest 2.0 PFLOPs/sec, doing mostly standard gridding, which has an AI around 24.5, the required bandwidth is 81.6 TBytes/sec, roughly three orders of magnitude higher than the average CSP output.

Without efficient caching of the visibility data and other operands required to perform gridding, 81.6 TBytes/sec of aggregated bandwidth will be difficult to achieve. The output bandwidth of a single NVMe SSD is around 2 GBytes/sec, so 40,000 disks would be required to achieve the required rate, and this is only at 2 PFLOPs/sec! On the other hand, if it would be possible to cache all the visibilities for a major cycle without having to read them again from disk, a reduction close to the number of iterations would be possible in the bandwidth. For our example, this would mean reducing the bandwidth to 4.0 TBytes/sec, which although still high, it looks more feasible. (At 1,000 parallel disks, it would be in the same order that filesystems implemented in other supercomputing facilities. For example, the "Fast Scratch" in Frontera supercomputer facility[1] achieves 1.5 TBytes/sec.)

In order to perform a more careful estimation of the required bandwidth, it is necessary to consider the dependency of the AI with the (w, ν)-plane, i.e., equation 15 would need to be applied by w-plane and frequency, following the w-distribution for each science case, as we did before to estimate the computing load in [2]. The aggregated bandwidth is a critical parameter and depends on the implementation, so we suggest for DMS to specify the required aggregated bandwidth empirically instead of relying on a theoretical estimation.

[1] According to `https://tacc.utexas.edu/systems/frontera/`.

As pointed out before, in order for the processing queue to be stable, the processing system throughput needs to match the average incoming data rate. Considering the EOP computational load as an example, if we specify the processing system to an utilization of 80%, the system should support an average throughput of 75 PFLOPs/sec, with the peak being around 1350 PFLOPs/sec.

Given the difference in scale between the extreme cases in the EOP and the normal cases, which is 3 orders of magnitude, the latency could be quite high for the most demanding cases. Consider EOP case 51, which requires 991 PFLOPs/sec and has a fraction of 2%. Processing this case, which was observed during $365 \times 0.02 = 7.3$ days, will require $(991/75) \times 7.3 = 96.5$ days, or roughly 3 months to be reduced in the ideal case. Taking into account efficiencies, this could easily extend to years of processing.

This large latency may be a problem (although currently there is no latency requirements for the system besides triggered observations), so it may be required to size the system above the average (provided that it is possible to speed up the computation). This is an overprice to pay for latency.

This takes us to consider on-demand (Cloud) computing resources an alternative to an on-premises system. Even if the cost of on-demand resources is higher than on-premises, it may be more cost effective to use on-demand resources in certain circunstances, because on-premises resources are payed-for even if not used, while on-demand resources are only payed when used. Given the characteristics of our demand (a relatively low base case with extreme cases for a small time fraction), it will be almost certainly the case that these additional resources needed to improve latency should be on-demand. We provide a simple analysis below, based on [4].

Figure 2 shows the demand distribution for the EOP. The green area above the average line (black horizonal line in the plot) fits into the white area below the average (these areas don't seem to match because of the log scale, but they do). We size the system for a given utilization ($\rho = 0.8$ in the plot), which is represented by the blue line. If the cost of on-demand resources is less than on-premises (and assuming that we can find a Cloud provider that can guarantee the availability of the resources required, and that the system complies with our technical requirements, which as we have seen demands a large number of GPUs and a very high bandwidth), then all the system resources should be procured on-demand. If, on the other hand, the on-premises cost is less than the on-demand cost for these resources, it will be more cost-effective to build the system on-premises. The criteria is simple in this case, but note that the cost comparison needs to include all the operational costs (in other words, the Total Cost of Ownership). For resources with a high utilization it is usually the case that it is less expensive to own the system. The rule of thumb is "own your base but contract the peak".

On the other hand, let's assume that we decide to over-size the system to reduce the maximum latency, and we explore the possibility of using Cloud resources for this over-capacity. The red segmented line represents the level of over-capacity that we build on premises. The green area above this line is the capacity that we allocate on on-demand Cloud resources. The white area between the red segmented line and the blue line represents FLOPs that are payed for if we build this over-capacity on premises, but that are not actually used. It is this on-premises resource under-utilization that can make on-demand processing atractive *even if the on-demand cost is higher than on-premises*, up to a certain extent.

Let's simplify the demand distribution as shown in Figure 3 to understand what should be the proportion of on-premises and on-demand resources. We define the cost of on-premises resources as c_r, the cost of on-demand resources as c_o and the premiun for using

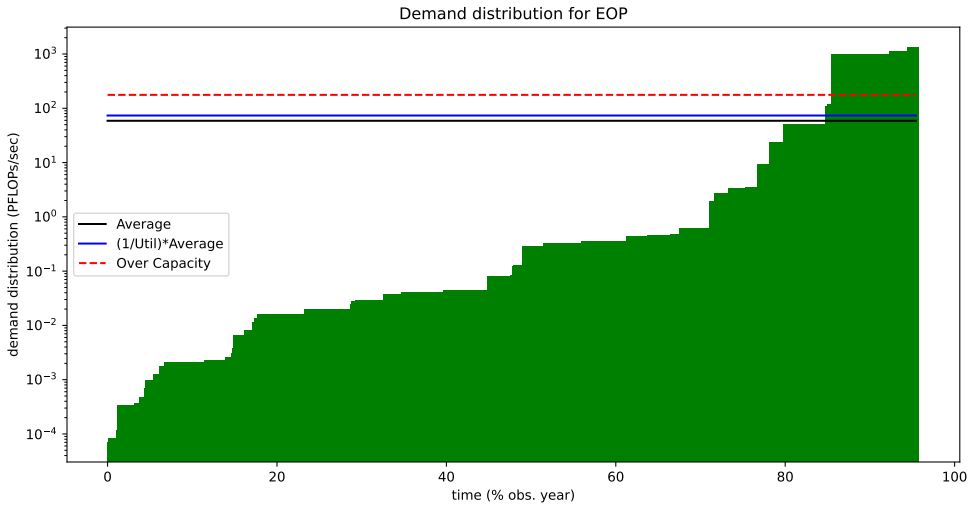

Figure 2: Demand distribution for the EOP.

on-demand resources as $U = c_o/c_r$. Using the parameters defined in Figure 3 the total cost is:

$$\begin{aligned} C &= FTc_r + VT_pUc_r \\ &= (P - V)Tc_r + VT_pUc_r \\ &= PTc_r + (T_pU - T)Vc_r \end{aligned}$$

If $T_p/T > 1/U$ then it is more cost effective to contract all the over-capacity as Cloud resources. If $T_p/T < 1/U$ then the over-capacity should be built into the on-premises system. Other solutions are possible depending on the demand distribution, in our case the decision is either on-demand or on-premises without an intermediate level for V due to the shape of our demand curve. Note that it is not necessary that $U < 1$ for on-demand resources to be preferrable.

Estimating Q_{proc} is complicated because it depends on several operational decisions and parameters. In theory, we could use equation 3, but this relation would severely underestimate the required size of this buffer, as this relation is based on the assumption that the visibilities leave the buffer as soon as a processor becomes available. In reality, the entire dataset for an Execution Block remains in the buffer for the entire time that it takes to process it and perform QA. This is an inefficient use of the processing buffer, as visibilities may sit there for long periods of time without being read. The design should evaluate better ways to use the processing buffer, as the cost of high-performance storage can be quite substantial. The size of the processing buffer depends on the latency for each observation, which as discussed before depends on the number of processing resources assigned to each job. This in turn depends on the scalability curves.

Figure 4 illustrates the problem, which is just an application of Little's law. (Little's law is stochastic though, and it's valid for any distribution of input rates and service times. Here we simplify things just looking at the problem deterministically.) These plots show the cumulative resources used as a function of time. In plot (a) new jobs (e.g. SBs) arrive with input rate λ to be processed. The yellow area can be thought to be all the FLOPs that

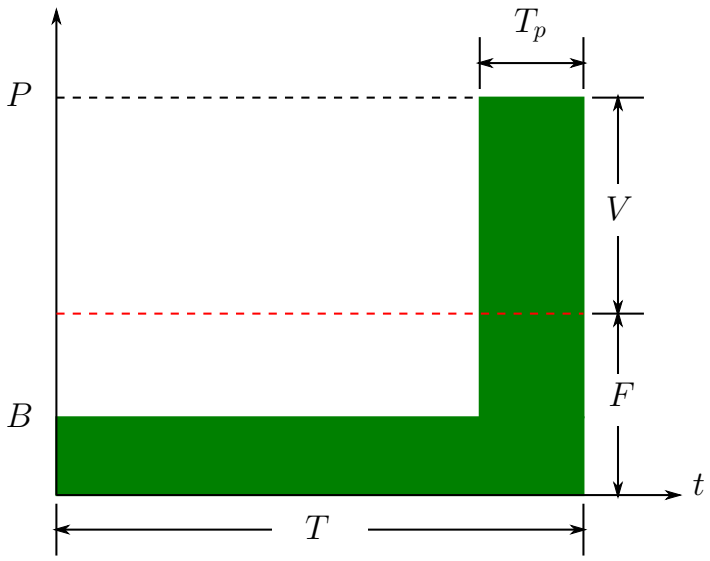

Figure 3: A simplification of the demand distribution from Figure 2.

need to be processed in time. We assign to each job all the capacity of the cluster, which in average will match the input data rate in FLOPs/sec. Hence, the jobs are processed right away, at the same rate that they are coming into the processing system. The total number of concurrent jobs at any given time is 1 (or close to 1 once we factor in the fluctuations in the data rate using equation 3). However, this **doesn't work at all for our case,** because the algorithms are iterative, so it is not possible to process all the FLOPs at once. Some FLOPs need to wait for other FLOPs to be processed. Therefore, we cannot assign all the cluster resources to one job, only a fraction of them. This is illustrated in plot (b) as decreasing the height of each job yellow bar. However, given that the area of each bar need to stay the same (all the FLOPs need to be processed anyhow), then the latency increases proportionally to R'. Increasing the latency means that now there are more jobs being processed concurrently, and as all the bytes of these jobs need to be on-disk for the duration of the processing, the size of the processing buffer needs to be larger. If the bytes for each job also need to stay in the processing buffer while quality assurance is performed, then the latency increases even more. If the behaviour of the parallelization speedup is such that it saturates at an even lower number of processors per job, then the latency increases even more and the size of the processor buffer increases proportionally.

This can be a significative cost. Let's see some numbers. Assuming that the length of each SB is around 4 hours, in the ideal case (plot (a)) it would take 4 hours of processing time to complete each job. However, assuming that the number of iterations is around 20, and that it will take around 3 days to perform QA, which is $3 \times 24/4 = 18 \approx 20$ additional SB durations in the latency, we assume that the latency is expanded by a factor of 40. We now have 40 jobs that need to be processed concurrently in the processing system. At an average data rate of 20 GBytes/sec, the volume of bytes is $40 \times 4 \times 3600 \times 20 = 11,520,000$ GBytes, or 11.5 PBytes. At a cost of 0.4 \$/Gbyte for SSD storage this is \$4.6MM. If the latency need to be expanded more, then the cost of the processing buffer increases proportionally. The actual size of the buffer could be an order of magnitude more when all details are taken into account (e.g., re-processing, high-data rate cases, extra processing

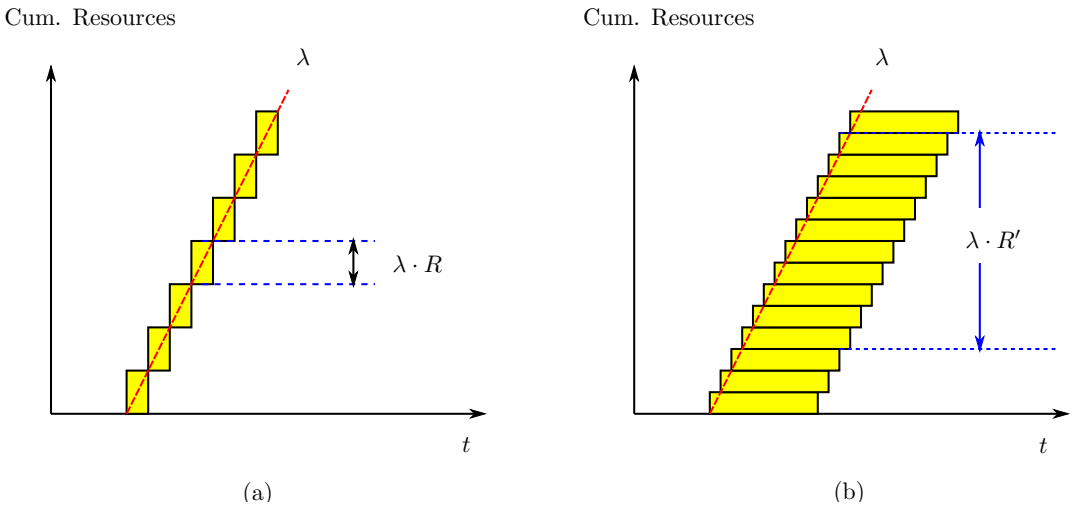

Figure 4: Relationship between the size of the processing buffer Q_{proc} and the latency.

space needed by each SB).

5 The archiving system

The archiving system is relatively simple, with one queue handling the incoming raw-data and product streams, and a processor that "ingest" the data into the Science Archive. The operations involved in this ingestion include gathering and writing the metadata into a database, storing the files into their final repository, and managing the replication of data. If λ_{sdc} is sized to be smaller than the peak value, then the fluctuations in the incoming data rates to the archive are "dampened", with the transport buffer absorving the fluctuations. Streams out of the archive include the movement of data to the processing system, end-user access and retrieval of the data products, and data replication. The detailed design of these systems (the Science Data Archive (SDA) and the Science Interfaces and Tools systems (SIT)) will probably incorporate their own additional internal buffers.

It can be assumed that newer datasets will be accessed more often than older datasets, so a hierarchical storage system where fast but expensive storage is used to store a sliding time window containing the most recent datasets, while older datasets are stored on slow but cheaper storage technologies would be a cost-efficient solution. For example, the cost of tape systems is around 0.008 $/GByte, while HDD storage is around 0.025 $/GByte. (These prices change with time, of course, and the total cost of ownership should be estimated for the ngVLA budget, not just the cost of the storage units. The cost projection should be studied by the specialists, here I'm just doing a quick search in the Internet to get some idea of the values.)

Maybe the most significant advantage of a tape system, besides being less expensive than hard disks, is that it doesn't consume power while the data is not being accessed. Estimating the power consumption of active storage to be between 0.3 and 0.7 Watts/GByte, at a data rate of 40.0 PetaBytes/month, the ngVLA would need to add between 12.0 and 28.0 MWatts of power each month, which is unfeasible. Storing the data into passive storage is necessary for ngVLA.

As noted before, all queues in the system need to comply with the basic stability criteria where the throughput must be greater than the incoming data rate. This means that writing to the tape system needs to sustain a data rate of 23 GBytes/sec (21 GBytes/sec for the EOP plus 10% for the data products), which may be challenging for a single tape system to achieve. A parallel tape system will probably be necessary.

In the case of the archiving system, the system can be modeled as a simple queue. Bytes arrive to the buffer and can be deleted as soon as they have been written to tape. Under this condition, equation 3 applies and the size of the buffer is relatively small. In addition, if X_{sdc} is not sized to the peak data rate (so the size of the transport buffer is non-zero), then fluctuations in the data rate are dampened by the transport buffer. If X_{sdc} is sized to the peak data rate, then the archiving buffer plays a similar role than the transport buffer when X_{sdc} is not sized to the peak, i.e., it needs to be sized as shown in section 3, with the bandwidth to the tape system replacing X_{sdc}.

6 Conclusions

A parametric model was developed for the transport, processing, and archiving systems. For the transport system we explored the tradeoff between the size of the transport buffer that receives the visibilities from the CBE, and the bandwidth of the WAN connection between the Central Electronics Building and the Science Data Center. The decision of whether to procure a network connection sized to the peak, or enough storage to stand the peak visibility data rate during a certain duration depends on the relative cost of networking versus storage and the duration of time that the system must stand the peak data rate. We develop a criteria that informs this decision.

We developed a simple model for the processing system, incorporating the iterative nature of the imaging algorithm as a re-circulation of visibilities. We discussed the relationship between the number of processing jobs in the system and their latency (Little's law). If the number of resources assigned to each job is decreased, then the latency and the number of concurrent jobs increases proportionally, with the result that the number of resources needed to process the input data rate stays constant. The same relationship indicates that if the entire dataset for the execution blocks are held on disk for the entire duration of the imaging and QA processes, the size of the processing buffer may need to be quite large (tens to hundreds of PBytes). We suggest to look for design alternatives to optimize the utilization of the processing buffer. We develop a relationship for the aggregated bandwidth (equation 15) and find it to be very challenging, a technical risk that is discussed below. We explored the possibility of using Cloud computing resources in the processing system and found a simple expression to guide this decision, based on the processing demand curve for the EOP. For the archiving system, we concluded that a tape system for long-term archival is required, because of the cost of a disk-based system and the electrical power that it would demand.

We identify the following technical risks:

- **Extreme required aggregated bandwidth [Scope, Cost].** A very large bandwidth will be required to feed into the computing processors the data operands needed to perform their computations at the rate required to sustain the average throughput (see equation 15). This problem can't be solved by processing jobs "slowly", because as it has been shown, increasing the latency just increases the number of jobs that need to be processed concurrently, with the result that the number of resources —

both the number of processors and the required bandwidth — stays the same. This is why this is not a cost problem, but a feasibility problem. If the required average bandwidth cannot be achieved, then the system won't be able to match the required processing throughput, and therefore not all science cases specified in the ROP/EOP will be able to be processed. A combination of efficient caching and large numbers of disks working on parallel (there's no other way of achieving high bandwidths from filesystems) could offer a feasible path. A high-bandwidth, high-capacity filesystem of this magnitude may pose technical challenges and a very high cost. Increasing the level of on-memory caching is not an easy problem either. We suggest as a mitigation strategy that this technical challenge be thoroughly analyzed and a realistic technical solution devised, incorporating algorithmic, infrastructural and architectural points of view.

- **Feasibility of avoiding the need to load CF from disk [Cost, Scope].** The aggregated bandwidth mentioned above was estimated under the assumption that the Convolution Functions are **not** loaded from disk storage. If they are, then the required bandwidth becomes even higher. It is important to validate this assumption. The feasibility of caching the CF, or computing them on-the-fly should be demonstrated.

- **Excessive latency due to poor parallelization efficiency [Cost, Scope].** Besides being an important cost driver, the parallelization efficiency could affect the latency that it is possible to achieve when imaging observations, because it could limit the number of processors that can be assigned to process each observation. If this ends up being a low number, very high latencies could result for observations that generate high computational loads (several months to process each observation). This is another reason to prioritize obtaining a realistic estimate of the parallelization efficiency. This could also affect the ability of the system to meet latency requirements for triggered observations.

- **High cost of processing buffer [Cost, Scope].** The size of the processing buffer could end up being an important cost driver for the processing system, and even more so if we require high bandwidth as well. The optimization of this system basically requires avoiding that bytes buffered there are not accessed for long periods of time.

References

[1] Neil J. Gunther. *Analyzing Computer System Performance with Perl: PDQ*. Springer Publishing Company, Incorporated, 2nd edition, 2011.

[2] R. Hiriart. ngVLA Computing Memo 11, ngvla Data Rates and Computational Loads (Update). Technical report, NRAO ngVLA, 2024.

[3] K. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Wiley, Chichester, 2001.

[4] Joe Weinman. Mathematical proof of the inevitability of cloud computing. `https://cloud-native-computing.de/materials/Joe_Weinman_Inevitability_Of_Cloud.pdf`, 2011. Accessed: 2024-05-29.

[5] Ginger Woolridge. Wide area networking 2022 pricing guide. https://lightyear.ai/resources/wan-connectivity-pricing-guide-p2p-mpls-dark-fiber-and-more, 2021. Accessed: 2024-05-29.