An Operational Analysis Primer
Part 1: System-level Evaluation

Tom Wilson

1 Introduction

On my recent project ([Wil14]), I was faced with the challenge of working with two groups of people who did not have significant experience with system performance evaluation. The first group consisted of the stakeholders (e.g., the customers). The second group consisted of my teammates, many of whom were inexperienced in all areas. I found myself having to teach them the basics of performance evaluation so that they could better understand the test results. I wish I had this primer already, but found myself writing a simpler version of this paper. Essentially, anyone involved with performance evaluation or capacity planning could benefit from this primer.

Operational analysis is the evaluation of system performance using measurement of parameters and established relationships between those parameters. It is also used to model performance. Operational analysis was formally introduced in [Buz76]. The most thorough source on the topic is [LZGS84]. Other sources are [Jai91], [Kan92], and [Gun05]. You might find the terms *operations analysis* and *operations research* in the literature, but these are different concepts that usually deal with the efficiency of business operations.

Operational analysis is often identified as part of queueing theory. While there are similarities between the two fields, they are distinct and independent. Operational analysis has been used to derive many results that queueing theory has also derived, but the assumptions and the means to the results are different.

Operational analysis leverages statistical means, or averages. This provides both strength and weakness. The strength comes through summarizing a large amount of data through a single number or a few numbers. The weakness comes in the form of loss of information (by summarizing) and the fact that means are influenced by outliers. However, the weakness can be overcome while retaining the strength.

This primer is divided into 4 parts.

- In part 1, I will provide most of the definitions and focus on system-level evaluation using operational analysis. I will expand on the material typically found in textbooks. One key area involves the instrumentation necessary to capture measurements.
- In part 2, I will describe resource-level evaluation. Attention will be given to obtaining appropriate measurements when subsystems and components are commercial-off-the-shelf.
- In part 3, I will discuss how operational analysis is used for modeling.
- In part 4, I will investigate some other topics that are not found in the literature.[1]

2 Basic Definitions

A general definition for a *system* is a set of interacting or interdependent components forming an integrated whole. I will constrain that slightly to computing systems or business-related systems. You should be familiar with what computing systems are. A grocery store and a manufacturing plant are examples of business-related systems. Operational analysis is valuable for all such systems.

A *resource* is a component of a system that performs work of some kind. For a grocery store, this could be a cashier. For a manufacturing plant, this could be a machine that assembles some parts. For a computer, this could be a processor.

Many resources in systems have a means of storing work to be done. This is generally called a *queue*. The resources in a system have some type of connectivity called a *network*. This should not be confused with concepts like a *Local*

[1] Parts 2-4 are planned to appear in future editions of *MeasureIT*.

Area Network (LAN) or *Wide Area Network* (WAN) because this network is internal to the system rather than one that connects the system to other systems.

A system receives inputs in the form of *requests*. Other terms for a request could be used: user, transaction, message, customer, or job. A system typically has outputs as well.

3 Parameters and Relationships

Operational analysis uses *parameters*, or *variables*, to describe the system. Some parameters are observed and some are derived ([FD90]). An *observed* parameter is obtained by watching the system function and measuring such parameters. Observed parameters typically count things (e.g., the number of requests) or capture interval lengths (e.g., elapsed time). A *derived* parameter is computed from observed parameters or other derived parameters. Not all references distinguish between observed and derived parameters. Parameters can be defined for system and resource levels. Most parameters exist at both levels. I will deal with resource-level parameters in part 2.

The relationship between a derived parameter and other parameters is simply an equation. If a relationship is always true, it is called a *law*. If a relationship is true when an assumption is made, it is called a *theorem* ([FD90]). One common assumption is *flow-balance*. I will deal with assumptions and theorems in part 3. Not all references distinguish between laws and theorems.

Table 1defines the common parameters and some laws. I will use the "$\equiv$" notation used in [LZGS84] to denote that a relationship exists by definition (these are laws by default). Some units are given as *seconds*, but they could be expressed in some related unit, like *minutes*. Other units are expressed as *requests*, but they could be any name given earlier (e.g., transactions).

Table 1: System-level Operational Analysis Parameters and Laws

Type	Parameter & Description		Laws	Units
Observed	T	Time Interval		Seconds
	A	Number of Arrivals		Requests
	C	Number of Completions		Requests
	W	Time in the System		Requests-Seconds
	B	Busy Time		Seconds
Derived	N	Number of Requests	$\equiv W/T = X * R$	Requests
	λ	Arrival Rate	$\equiv A/T$	Requests/Second
	X	Throughput	$\equiv C/T$	Requests/Second
	S	Service Time	$\equiv B/C$	Seconds
	R	Response Time	$\equiv W/C$	Seconds
	U	Utilization	$\equiv B/T = X * S$	-

System parameters are usually denoted by a symbol with no subscript (e.g., X). Resource parameters are denoted by a symbol with a subscript (e.g., X_k), where the subscript indicates the resource number. Some literature uses the subscript 0 for system parameters (e.g., X_0). I will not use a subscript for system parameters. Also, some literature uses different symbols for the parameters, which can make things confusing.

3.1 Observed Parameters

Figure 1 shows a simple view of a system. It has a number of arrivals (A) and a number of completions (C). These only have meaning for a measurement (time) interval (T). As time passes and arrivals occur, the time in the system (W) and the busy time (B) are observed. More specifically, they are measured via some instrumentation, and this instrumentation need not be inside the system. Real systems might have multiple input and output points and multiple servers, each with its own queue.

Each observed parameter will be discussed in detail with reference to Figure 2. In the figure, each box represents a request, with the duration of the execution indicated as a value in the box (note that we do not need to know the execution duration of each request). For convenience, all times have a one-second resolution. I will also focus on the instrumentation necessary to measure each parameter.

$\boldsymbol{T}$: The time interval is the period when observations are made, or when measurements are taken. In Figure 2, T is 50, corresponding to the interval [0,50]. Since T is only a value, we lose a lot of important information about the interval. If we look at a system at two different times that have the same duration, it is quite likely that we will get different

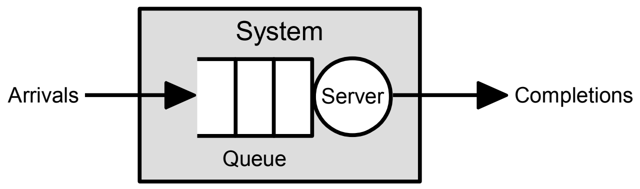


Figure 1: The system view in operational analysis ignores the internal details of the system and focuses on external arrivals and completions during the measurement interval.

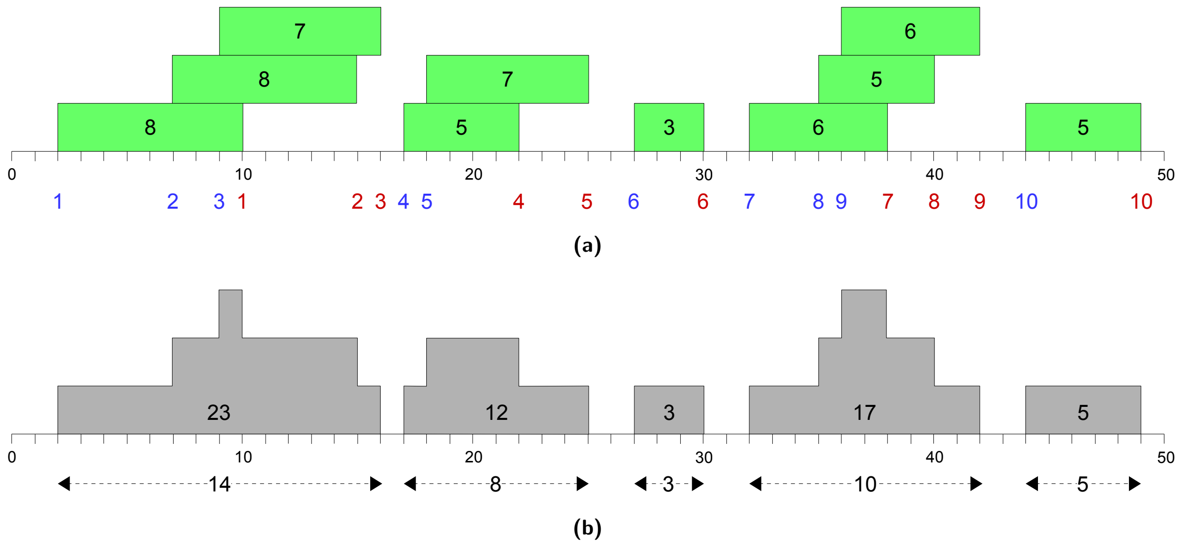


Figure 2: Example 1: **(a)** illustrates the time interval (T), the number of arrivals (A) using blue text, and the number of completions (C) using red text. **(b)** illustrates the time in the system (W) using grey polygons and the busy time (B) using dashed arrows.

measurements for other parameters. Commonly, the inputs to the system are different (in content and/or timing). The instrumentation for T is simply some means to capture the start and end times and compute the difference.

$\boldsymbol{A}$: The number of arrivals is a count of the requests that begin during the time interval. Figure 2(a) highlights the number of arrivals using blue numbers below each arrival. The instrumentation to capture the number of arrivals is simply a counter that increments as each arrival occurs. By only counting the arrivals, we lose the information concerning when they arrived. We also do not know what the arrivals actually were.

$\boldsymbol{C}$: The number of completions is a count of the requests that end during the time interval. Figure 2(a) highlights the number of completions using red numbers below each completion. Here, the sequence of completions is the same as the sequence of arrivals, but this does not need to be the case. The instrumentation to capture the number of completions is a counter that increments as each completion occurs.

$\boldsymbol{W}$: The time in the system is the amount of time that all requests spend in the system. As Table 1 indicates, this is an *area* measurement with units of *requests-seconds*. Figure 2(b) illustrates W using the grey polygons, with the contribution toward W shown by the number in each polygon. This is formed by a count of messages in the system for each time unit (since we have a one-second resolution).

$\boldsymbol{B}$: The busy time is the amount of time that the system is processing one or more requests. Figure 2(b) illustrates B using the arrows below the polygons, with the contribution toward B shown by the number within each arrow.

Now, let's look at computing W in more detail. Table 2 shows the completion and arrival times. Below these times are the differences between the times. If we sum these differences, we get 60. Also, if we sum all of the completion times and subtract the sum of all of the arrival times, we also get 60.

What the table is computing is captured by Equation 1, where t_{c_i} is the time of completion i, t_{a_i} is the time of arrival i, and n is the number of requests. Put more simply, we can do the following: Every time an arrival occurs, subtract the arrival time from W; every time a completion occurs, add the completion time to W. We will discuss problem cases where

Table 2: Computing W for Example 1

Event	Time										Total
Completion	10	15	16	22	25	30	38	40	42	49	287
Arrival	2	7	9	17	18	27	32	35	36	44	227
Difference	8	8	7	5	7	3	6	5	6	5	60

requests overlap the time interval boundaries later.

$$W = \sum_{i=1}^{n}(t_{c_i} - t_{a_i}) = \sum_{i=1}^{n} t_{c_i} - \sum_{i=1}^{n} t_{a_i} \tag{1}$$

Let's look at computing B in more detail. Table 3 shows the times for the arrivals and completions, the event (i.e., arrival or completion), and the count of requests within the system. We denote when an arrival increments the count from 0 to 1 and when a completion decrements the count from 1 to 0 by vertical lines in the table. When we subtract the former time from the latter, we arrive at the amount of time that the system is busy.

Table 3: Determining Busy Periods for Example 1

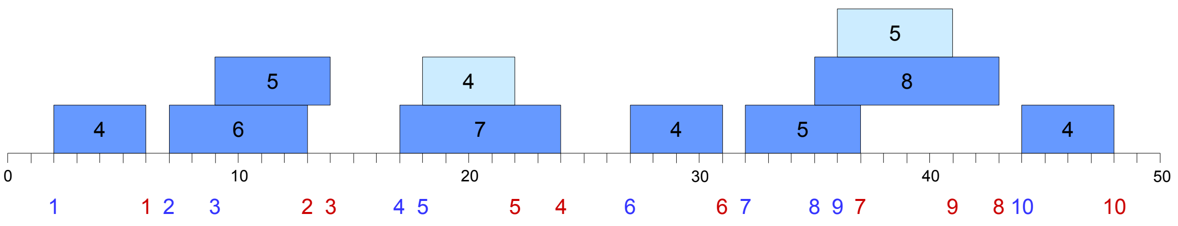
Time	2	7	9	10	15	16	17	18	22	25	27	30	32	35	36	38	40	42	44	49
Event	A	A	A	C	C	C	A	A	C	C	A	C	A	A	A	C	C	C	A	C
Count	1	2	3	2	1	0	1	2	1	0	1	0	1	2	3	2	1	0	1	0

For clarity, I will demonstrate one calculation. The first arrival at time 2 marks the beginning of the first busy period. The third completion at time 16 marks the end of the first busy period. The difference in time is 14 seconds. Table 4 summarizes the computations. This is illustrated in Figure 2(b) by the dashed arrows below the time interval.

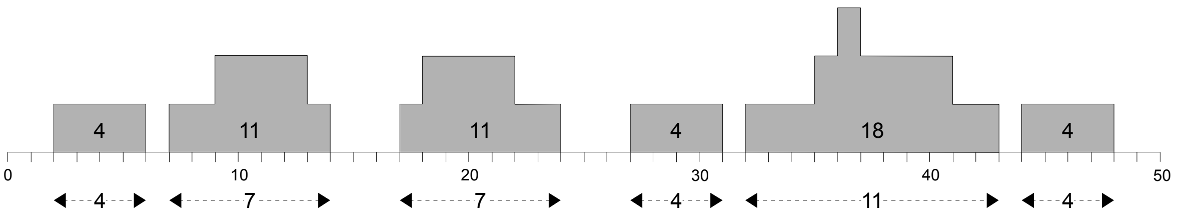
Table 4: Computing B for Example 1

Event	Time					Total
Last Completion	16	25	30	42	49	162
First Arrival	2	17	27	32	44	122
Difference	14	8	3	10	5	40

Figure 3(a) shows another example interval with arrivals and completions. The arrival times are the same as those in Figure 2, but the completion times are different. For the two requests that are colored light blue, the requests arrive and complete during other requests. This is not an issue, but provides contrast to the previous example.



(a)



(b)

Figure 3: Example 2 has the same arrival times as Example 1, but the completion times are different.

Table 5 shows the computation for W. The arrival and completion times pair up for most requests. However, requests 4, 5, 8, and 9 have times which are colored, green, purple, blue, and red, respectively. So, as W is being computed on-the-fly, the arrivals and completions are being processed as they occur. The result is the same when compared to pairing up the times by request (e.g., (22 - 17) + (24 - 18) = (24 - 17) + (22 - 18)).

Table 5: Computing W for Example 2

Event	Time										Total
Completion	6	13	14	22	24	31	37	41	43	48	279
Arrival	2	7	9	17	18	27	32	35	36	44	227
Difference	4	6	5	5	6	4	5	6	7	4	52

We will compute B from Figure 3: $B = 4 + 7 + 7 + 4 + 11 + 4 = 37$.

3.2 Derived Parameters

Each derived parameter will be briefly discussed with reference to Table 6. While T, A, and C are the same for both examples, other observed parameters are not.

Table 6: Operational Analysis Summaries for Examples 1 & 2

	Observed					Derived					
Example	T	A	C	W	B	N	λ	X	S	R	U
1	50	10	10	60	40	1.20	0.2	0.2	4.0	6.0	80%
2	50	10	10	52	37	1.04	0.2	0.2	3.7	5.2	74%

$\boldsymbol{N}$: The number of requests in the system is the average of the number of requests in the system over the time interval. This is computed from W/T. Because each example has a different W, N is also different. The first example has more requests inside the system during the time interval. Because the number of arrivals is the same, the requests must be in the system longer. Since N is an average, it might not be an integer.

The equation $N = X * R$ is known as *Little's Law* ([Lit61]). It really appears as $N = \lambda * R$, but the theorem $X = \lambda$ allows X to be substituted. Most books present a proof of the law, but I will not bother since we are more interested in applying these concepts rather than reiterating established theory. I should note that this law precedes the establishment of operational analysis by 15 years. This is because the law was a result of queueing theory research, but it is still applicable to operational analysis.

$\boldsymbol{\lambda}$: The arrival rate is simply A/T. Both examples have the same arrival rate since they have the same number of arrivals. An average arrival rate suggests uniform arrivals, but this is typically not the case.

$\boldsymbol{X}$: The throughput is simply C/T. Both examples have the same throughput since they have the same number of completions. We would expect X to be the same as λ—obviously, when $C = A$, it is. We will discuss situations where it is not later in this paper.

$\boldsymbol{S}$: The service time is B/C. The system services requests more quickly for the second example. Note that if there are no completions, then S is undefined.

$\boldsymbol{R}$: The response time is W/C. The system has a faster response time for the second example. Note that if there are no completions, then R is undefined. Also note that if we average the individual response times, we get the same results: for example 1, $(8+8+7+5+7+3+6+5+6+5)/10 = 6.0$; for example 2, $(4+6+5+7+4+4+5+8+5+4)/10 = 5.2$.

$\boldsymbol{U}$: The utilization is B/T. Obviously, because the system is busier in example 1, it has a higher utilization. A high utilization for a system should not necessarily be a concern, whereas a high utilization for a resource should be a concern. We generally want to know when a system can be fully utilized and still have good performance.

The equation $U = X * S$ is known as the *utilization law*.

We will create one more example to illustrate the impact of T. Figure 4 shows two examples where the time between two sets of arrivals is different.

Table 7 shows the values for the parameters for example 3. Obviously, N, λ, X, and U are all different since their equations involve T. However, the point of this example concerns how conclusions are drawn. It appears that the second example has a lower throughput (X). But it is because there is a lower arrival rate (λ). The performance of the system is essentially the same because the service time (S) and response time (R) are identical.

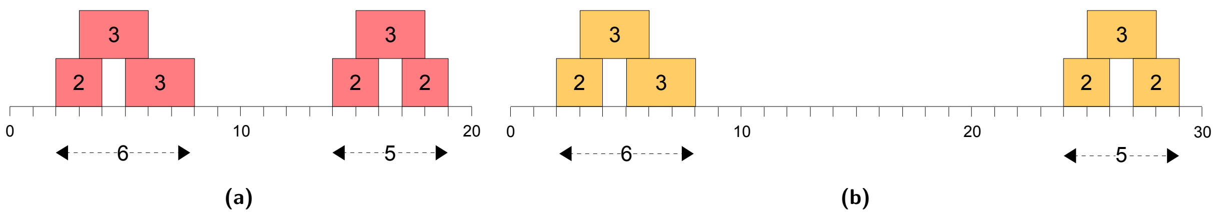


Figure 4: Example 3: **(a)** and **(b)** show similar arrivals and completions over different time intervals.

Table 7: Operational Analysis Summary for Example 3

	Observed					Derived					
Example	T	A	C	W	B	N	λ	X	S	R	U
3a	20	6	6	15	11	0.75	0.30	0.30	1.83	2.5	55.0%
3b	30	6	6	15	11	0.50	0.20	0.20	1.83	2.5	36.7%

4 Orphan Requests

Most sources do not discuss partial requests, although [Buz76] does. I will define a few new terms so as to avoid confusion in my discussion. An *orphan arrival* is a request where the arrival is in the time interval but the completion is not. An *orphan completion* is a request where the completion is in the time interval but the arrival is not. An *orphan request* is either an orphan arrival or an orphan completion. I will denote the number of orphan arrivals by A^o and the number of orphan completions by C^o.

Figure 5(a) shows an excerpt from Example 1 where time interval has changed. Here, 3 requests arrive, but only one completes during the time interval. So, there are two orphan arrivals: $A^o = 2$. Figure 5(b) shows how W and B are affected. When we compute the response time, we get a result that does not make sense: $R = W/C = 13/1 = 13$. Clearly, none of the requests take that long.

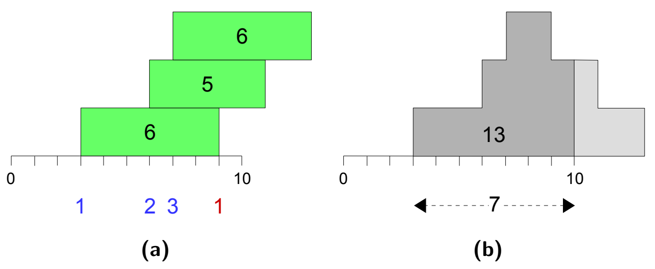


Figure 5: An example with orphan arrivals.

Figure 6(a) shows another excerpt from Example 1. Here, there are two orphan completions: $C^o = 2$. Figure 6(b) shows the impact to W and B. Here the response time is $R = 16/3 = 5.33$. No request takes that short a time.

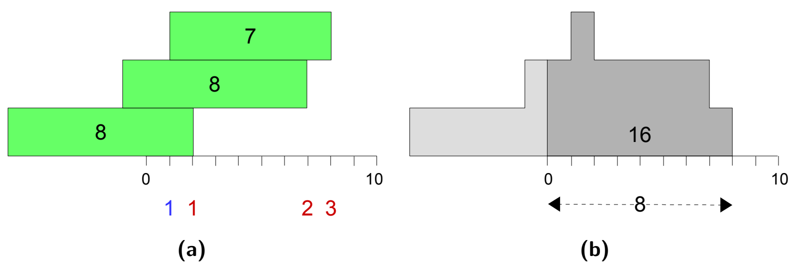


Figure 6: An example with orphan completions.

Consider a situation where the time interval in Figure 6 is followed by the time interval in Figure 5. In this case, there are both orphan arrivals and completions. The response time will be $R = (16 + 13)/4 = 7.25$. It is hard to decide if this value is correct even though $A = C$. If all of the requests were in the time interval, then the response time would be $R = 6.67$.

If $A^o + C^o > 0$ then R is likely to have error. Only when the orphan arrivals and the orphan completions mirror each other do the errors cancel out. The magnitude of the error appears to be influenced by (1) the magnitude of $(A^o + C^o)/C$ and (2) the magnitude of R/T. In the first case, when there are few orphans relative to the total number of completions, the error tends to be diminished by the impact of the non-orphan requests. In the second case, if the response time is small relative to the time interval, then the error tends to be smaller.

Instrumentation needs to handle orphans properly. When computing W, the number of requests in the system at the beginning of the time interval needs to be set to the number of requests already in the system. As completions occur, this number will decrement properly.

5 Bottlenecks

A *bottleneck* exists in a system if it has a resource that saturates (i.e., its utilization reaches 100%) as load increases. Essentially, as the arrival rate increases to the system, it will also increase to the resources. One resource will reach a point where its output rate cannot keep up with the input rate. Requests within the system get queued and wait for service. We will talk more about resource bottlenecks in part 2. This causes an impact at the system level by a reduction in throughput for an appropriate time interval around the time when the increased arrival rate occurs.

For a varying workload over a time interval, a bottleneck can occur and then dissipate without detection. This would be because the arrival rate during T is lower than the arrival rate that causes the bottleneck. However, for a short time within T, the arrival rate can rise beyond the threshold for the bottleneck to occur. Because T is long enough, the arrival rate will drop and the queued requests eventually get serviced. All of the arrivals complete and we do not notice anything concerning A, C, λ, or X. We should note a rise in R if we have multiple observations to compare.

Consider the example in Figure 7. The system is capable of processing a request in 3 seconds. Only one request can be handled at a time. Requests that arrive too quickly will get queued until the server can handle them. There are 9 arrivals and 8 completions, so $\lambda = 9/40 = 0.225$ and $X = 8/40 = 0.200$. While the λ and X are not equal, they are close and the orphan arrival that causes the discrepancy only exists because the red request arrives too close to the end of the time interval. The response time is $R = 38/8 = 4.75$. Note that the service time is $26/8 = 3.25$ rather than $27/9 = 3$ because of the orphan request.

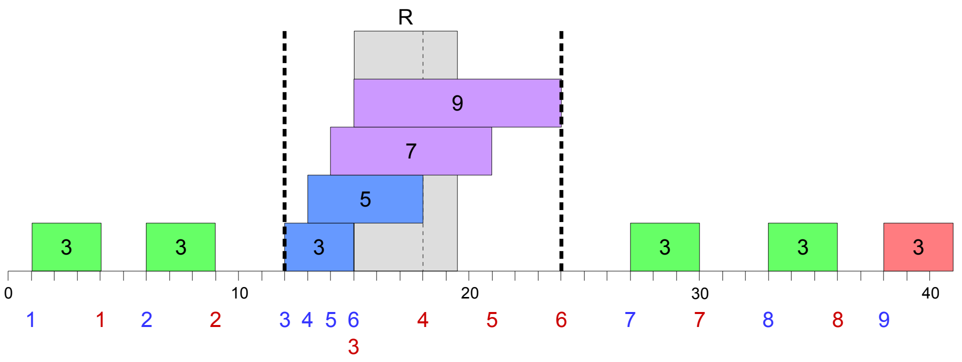


Figure 7: Bottleneck example.

Two situations cause orphan arrivals. The first is caused by arrivals that are too close to the end of the time interval. This would generally mean that the time between the arrivals and the end of the time interval is less than the response time. In this case, we mean a response time that is not inflated due to a bottleneck. The red request is this type of orphan.

The other case is very similar, but is caused by a higher-than-average response time. What if we change T to be a time containing a shorter interval when the higher arrival rate occurs? If we select the interval properly, we will have a throughput that is less than the arrival rate, which means that there are more arrivals than completions. In other words, we will have a situation where there are orphan arrivals.

The time frame is highlighted by the dashed vertical lines, containing the blue and purple requests. The grey rectangle denotes the time of the last request until the average response time (4.75) has passed. The dashed line within the rectangle denotes when the fixed processing time (3.0) has elapsed. We could use either 3.0 or 4.75 for the sake of argument, but the actual value is not important. Any completion past the rectangle is an orphan because of the bottleneck rather than an orphan because the arrival is too close to the end of the interval under normal conditions. These are the purple requests. If requests continued to arrive at a high rate (here, it is 1.0), the throughput would still be throttled at $1/3$.

Now, consider when T is 20 for the interval [0,20]. Now we $A^o = 2$. The two orphans are arrivals 5 and 6. The two missing completions are because of the bottleneck, not because of the late arrival time (like arrival 9 in the original example). Arrival 6 is too far away from the end of the interval in an idle system. Of course, we are dealing with a simple example. Real requests have different response times and it is harder to decide which ones are too close to the end of the interval.

In order to see hidden bottlenecks, the time interval may need to be divided into smaller intervals. A high response time for a smaller interval is a good indicator of bottleneck presence. I will illustrate this situation in a future paper unrelated to this primer.

When U is not 100%, then the source of requests is a bottleneck. Because the system is idle some of the time, it could handle more requests. This does not mean that there is not also a bottleneck within the system during some of the busy periods as we just discussed. However, a system can be fully utilized and no bottleneck exists except for the input. A trivial example is a situation where requests arrive R seconds apart for the entire time interval with a response time of R. We would observe that $B = T$, and derive that $U = 100\%$ and $S = R$.

6 Conclusions

This paper has discussed system-level evaluation using operation analysis. Operational analysis is a well-established discipline that is very powerful. Subsequent parts will investigate operational analysis applied to the resource-level evaluation of a system and modeling.

The concepts discussed here should be used when developing performance tests. You should understand the impact of orphan requests on your results. Results can be hard to accept when the throughput is greater than the arrival rate! Your tests may have orphan completions when heavy loads exist. You need to justify that the throughput degradation is because of a bottleneck rather than an unfortunate arrival pattern.

Bibliography

[Buz76] J. P. Buzen. "Fundamental Operational Laws of Computer System Performance". *Acta Informatica*, 7(2):167–182, 1976.

[FD90] Paul L. Fortier and George R. Desrochers. *Modeling and Analysis of Local Area Networks*. CRC Press, 1990.

[Gun05] Neil Gunther. *Analyzing Computer System Performance with Perl::PDQ*. Springer, 1st edition, 2005.

[Jai91] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, Inc., 1991.

[Kan92] Krishna Kant. *Introduction to Computer System Performance Evaluation*. McGraw-Hill, 1992.

[Lit61] John D. C. Little. "A Proof of the Queuing Formula: L=λW". *Operations Research*, 9(3):383–387, 1961.

[LZGS84] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance, Computer System Analysis Using Queuing Network Models*. Prentice Hall, 1984.

[Wil14] Tom Wilson. "My Great Performance Testing Project". *CMG MeasureIT*, Issue 14.1, February 2014.