Guided Exercises*

Federico Maggi – fmaggi@elet.polimi.it

October 27, 2010

Contents

*From "Quantitative System Performance", Lazowska *et al.*, 1984

1 Notations

Variable	Definition
T	length of an observation interval
N	population
Z	average think time of a terminal user
X	system-level throughput
R	system response
X_k	throughput observed at k-th resource
U_k	utilization of resource k
S_k	service time per visit of resource k

Law	Definition
Utilization	$U_k = XS_k = XD_k$
Little's	$N_k = X_k R_k$
Response time	$R = \frac{N}{X} - Z$
Forced flow	$X_k = V_k X$

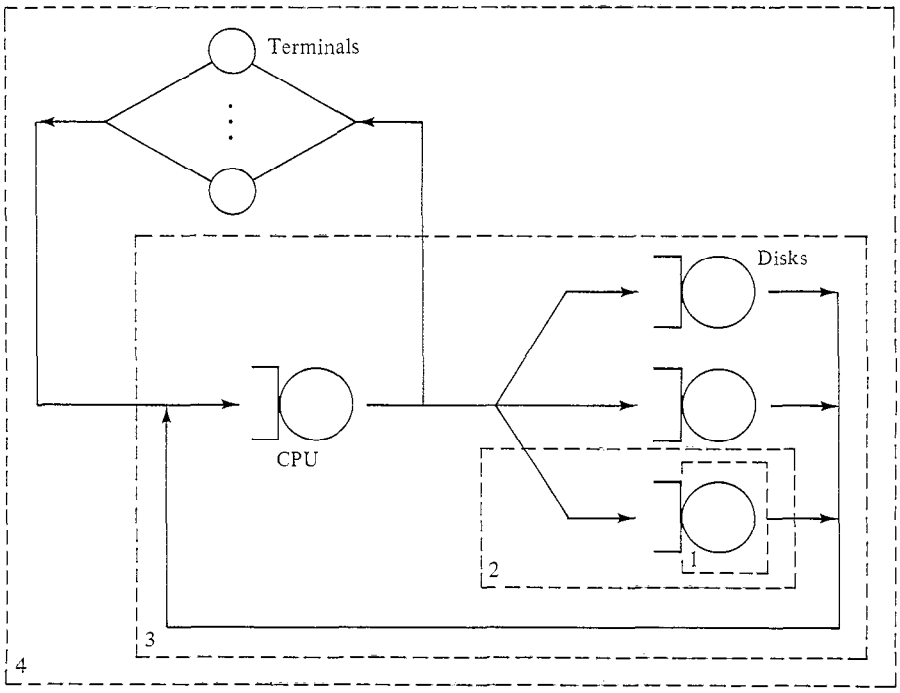

Figure 1: Little's applied at four levels.

2 Exercises

2.1 Little's Law and Forced Flow Law

Exercise 1

On the system depicted in Figure 1, the following measurements were made:

- throughput of each disk is 40 request/seconds ($X_{disk} = 40r/s$)
- 4 customers, on average, are either in queue or in service on each disk ($N_{disk} = 4$)
- average think time is 4 seconds ($Z = 5s$)
- the service time on each disk is 0.0225 seconds ($S_{disk} = 0.0225s$)

1. calculate the utilization of one disk without considering its queue.
2. calculate the average time spent by each request in each disk, both in queue and in service.
3. calculate the average number of users in service and the average number of users awaiting service.

4. calculate the number of interactions per second, considering that the system-level average response time is 15 seconds and there are 7.5 *active* (i.e., non-thinking) users.

5. calculate the total number of users.

Answer of exercise 1

Most of the questions can be answered by using Little's law in both its original formulation, as the utilization law, and as the response time law.

1. A disk without its queue (box 1) can only hold one job at a time, thus its utilization can be derived directly by using the utilization law:

$$\begin{aligned} U_{disk} &= X_{disk} S_{disk} \\ &= 40 \cdot 0.0225 \\ &= 0.9 \end{aligned}$$

2. The average time spent by each request in each disk is not S_{disk} anymore if the queue is taken into account. Thus we can apply Little's law and obtain:

$$\begin{aligned} R_{disk} &= \frac{N_{disk}}{X_{disk}} \\ &= \frac{4}{40} \\ &= 0.1s \end{aligned}$$

   Note that, since $S_{disk} = 0.0225$ (time spent in the disk), the time spent in queue is

$$t_{queue,disk} = R_{disk} - S_{disk} = 0.0775s$$

3. the number of users in queue and in service can be calculated directly from the disk utilization.
   - $U_{disk} = 0.9$ users are serviced into the disk
   - $N_{disk} = 4$ users are either serviced or in queue
   - $N_{queue,disk} = 4 - 0.9 = 3.1$

4. to apply Little's law at box 3, we need N and R, which are both given. Thus:

$$\begin{aligned} X &= \frac{N}{R} \\ &= \frac{7.5}{15} \\ &= 0.5i/s \end{aligned}$$

5. we need the response time law:

$$\begin{aligned} N &= X(R+Z) \\ &= 0.5(15+5) \\ &= 10 \end{aligned}$$

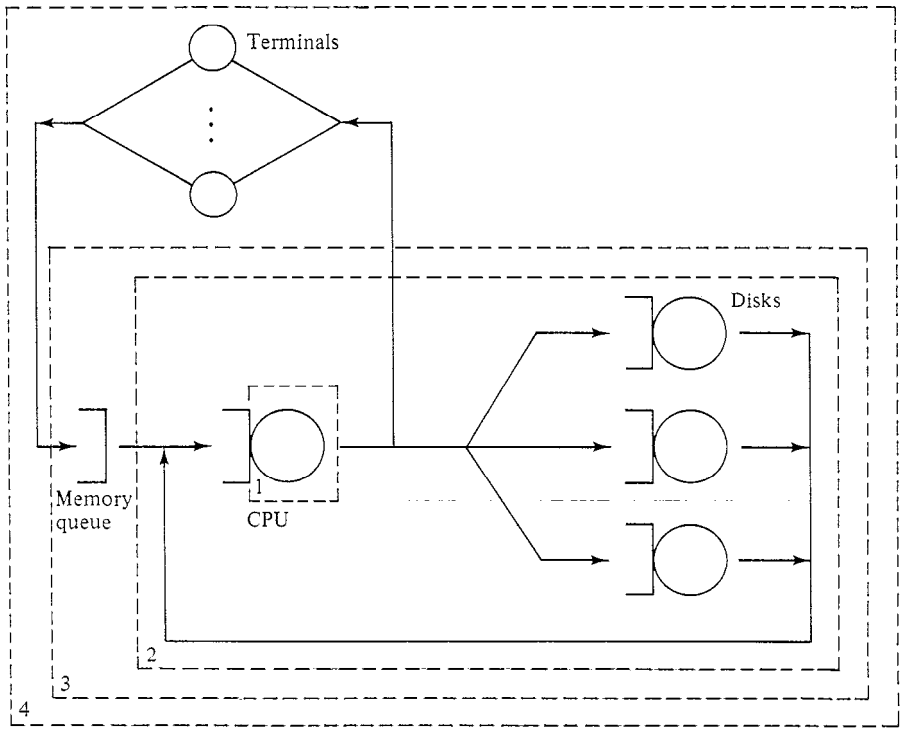

Figure 2: Little's applied to a memory-constrained system.

Exercise 2

On the system depicted in Figure 2, the following measurements were made:

- average number of users is 23 ($N = 23$)
- average response time as perceived by a user is 30 seconds ($R = 30s$)
- throughput is 0.45 interaction/second ($X = 0.45i/s$)
- average number of requests in memory is 1.9 ($N_{Box2} = N_{memory} = 1.9$)
- average CPU service demand per interaction is 0.63 seconds ($D_{cpu} = 0.63s$)

1. what is the average think time of a user?
2. how many users are attempting to obtain service and how many of them are instead thinking?
3. how much time elapses between the acquisition of memory and the completion of an interaction?
4. what is the utilization of the CPU?

Answer of exercise 2

Most of the questions can be answered by using Little's law in both its original formulation, as the utilization law, and as the response time law.

1. the average think time must be calculated at box 4, by using the response time law:

$$\begin{aligned} Z &= \frac{N}{X} - R \\ &= \frac{23}{0.45} - 30 \\ &= 21s \end{aligned}$$

2. the users that are *not* thinking are those who are attempting to access the memory, thus can be obtained by applying Little's law at box 3 (i.e., right before the queue).

$$\begin{aligned} N_{want\ memory} &= N_{Box3} \\ &= XR \\ &= 0.45 \cdot 30 \\ &= 13.5 \end{aligned}$$

   Since 13.5 users are in memory, either waiting or being serviced, and $N_{inmem} = 1.9$:

$$\begin{aligned} N_{queue,mem} &= N_{want\ memory} - N_{mem} \\ &= N_{Box3} - N_{Box2} \\ &= 13.5 - 1.9 \\ &= 11.9 \end{aligned}$$

   are in queue.

3. the time elapsed between the acquisition of memory (i.e., out from its queue) and the completion of an interaction is calculated by applying Little's law at box 2:

$$\begin{aligned} R_{memory} &= R_{Box2} \\ &= \frac{N_{memory}}{X} \\ &= \frac{N_{Box2}}{X} \\ &= \frac{1.9}{0.45} \\ &= 4.2s \end{aligned}$$

Interestingly, the time spent in queue (i.e., $R - R_{inmem} = 25.8s$) exceeds of one order of magnitude the time spent to complete an interaction (just $4.2s$).

4. the utilization of the CPU can be calculated by applying the utilization law at box 1. That is:

$$\begin{aligned} U_{CPU} &= XD_{CPU} \\ &= 0.45 \cdot 0.63 \\ &= 0.28 \end{aligned}$$

Interestingly, only a fraction of the CPU is utilized and, once again, the culprit is the slow memory queue.

2.2 Asymptotic bounds on systems with terminal workload

Exercise 3

Consider a web application deployment structured as follows:

- one web server (WS) that renders dynamic pages,
- one application server (AS) that constructs such dynamic pages,
- one database server (DB) that holds the data needed by the application server.

The web server is the most demanded center, it has indeed $D_{WS} = D_1 = 5ms$. The application server and the database server have a service demand of, respectively, $D_{AS} = D_2 = 4ms$ and $D_{DB} = D_3 = 3ms$.

Using a modified version of `ab`, the *Apache HTTP server benchmarking tool*[1], the web application is tested in a closed deployment with a terminal workload of 20 customers with a think time of 7 milliseconds.

1. Draw the queuing network model.
2. What is the *primary* bottleneck center?
3. What is the *secondary* bottleneck center?
4. What is the minimum response time that can be achieved?
5. One of the developers claims that its code, installed into the application server, would result in a system response time of $80ms$. Is he correct?
6. What is the effect of replacing the application server with a new one that is twice as faster?
7. Is it possible to increase the maximum throughput of this deployment without modifying the hardware configuration at all? Why?
8. What is the modification, if any, that would allow the application server to run at its maximum speed?

Answer of exercise 3

1. The diagram can be generated with the JMTGraph[2] tool, for example. However, it is quite irrelevant because we already know all the D_k, thus the layout of the network does not affect our calculations.

2. The *primary* bottleneck center limits the throughput to $X(N) \leq \frac{1}{D_{max}}$. In our case:

$$X_{max}(N) = \frac{1}{D_{max}} = \frac{1}{D_{WS}} = \frac{1}{5ms} = 200r/s$$

[1] The original version of `ab` only allow batch workload: `http://httpd.apache.org/docs/2.0/programs/ab.html`

[2] http://jmt.sf.net

3. The *secondary* bottleneck is the one that would become the primary bottleneck if the primary one is removed. In our case:

$$X_{max'}(N) = \frac{1}{D_{max'}} = \frac{1}{D_{AS}} = \frac{1}{4ms} = 250r/s$$

Obviously, $X_{max}(N) < X_{max'}(N)$ and thus, if we do not alleviate the bottleneck due to the web server, the system throughput is limited to 200 requests per second.

These are actually asymptotic bounds and can be derived by plotting $X(N)\frac{1}{D}$ and $X(N) = \frac{N}{D+Z}$ $\forall D \in \{D_{WS}, D_{AS}, D_{DB}\}$ as shown in Figure 3.

4. Intuitively, each request has to visit all the centers (with a corresponding number of visits that may change for each center). This information is summarized by means of the centers' response time, D_{WS}, D_{AS}, D_{DB}.

Thus, the lowest response time that can be achieved is simply D, where:

$$R(N) \geq D = \sum_{k \in \{WS,AS,DB\}} D_k = (3+4+5)ms = 12ms$$

This can be visualized by means of the bounds of the asymptotic bounds of $R(N)$, as show in Figure 4.

5. The claim of the developer is indeed correct, because the lower bound of $R(N)$ is $D = 12ms < 80ms$.

In particular, the web server causes the whole system to have $R(N) = 80ms$ when (without considering the think time):

$$R' = N'D_{max} - Z \Leftrightarrow N' = \frac{R' + Z}{D_{max}} = \frac{80ms}{5ms} = 16$$

6. Doubling the speed of a center means that its new service time is half of the old service time. Since we assume that the number of visits remains the same, the only variable that reflects this change is the service demand.

More formally:

$$D'_{AS} = V_{AS}S'_{AS} = V_{AS}\frac{S_{AS}}{2} = \frac{2ms}{2} = 1ms$$

A quick glance at the asymptotic bounds reveals that, unfortunately, the effect of replacing the application server is not worth the money. In fact, the application server is not limiting the throughput (so the response time) because it is the *secondary* bottleneck.

However, if $D_{AS'} < D_{AS}$ the effect is reflected in term of a lower system demand $D' < D$. As a consequence the light workload asymptote, $X(N) \geq \frac{N}{D+Z}$.

7. No, because we would at least need to replace the web server (bottleneck). Since the nature of the jobs of the web server (i.e., handling HTTP requests) is very different from the nature of the jobs of the application server, one can neither re-balance the load between web server and application server.

8. The modification consists in replacing the web server with one that has a $D'_{WS} \geq D_{AS}$, so the application server would become the primary bottleneck.

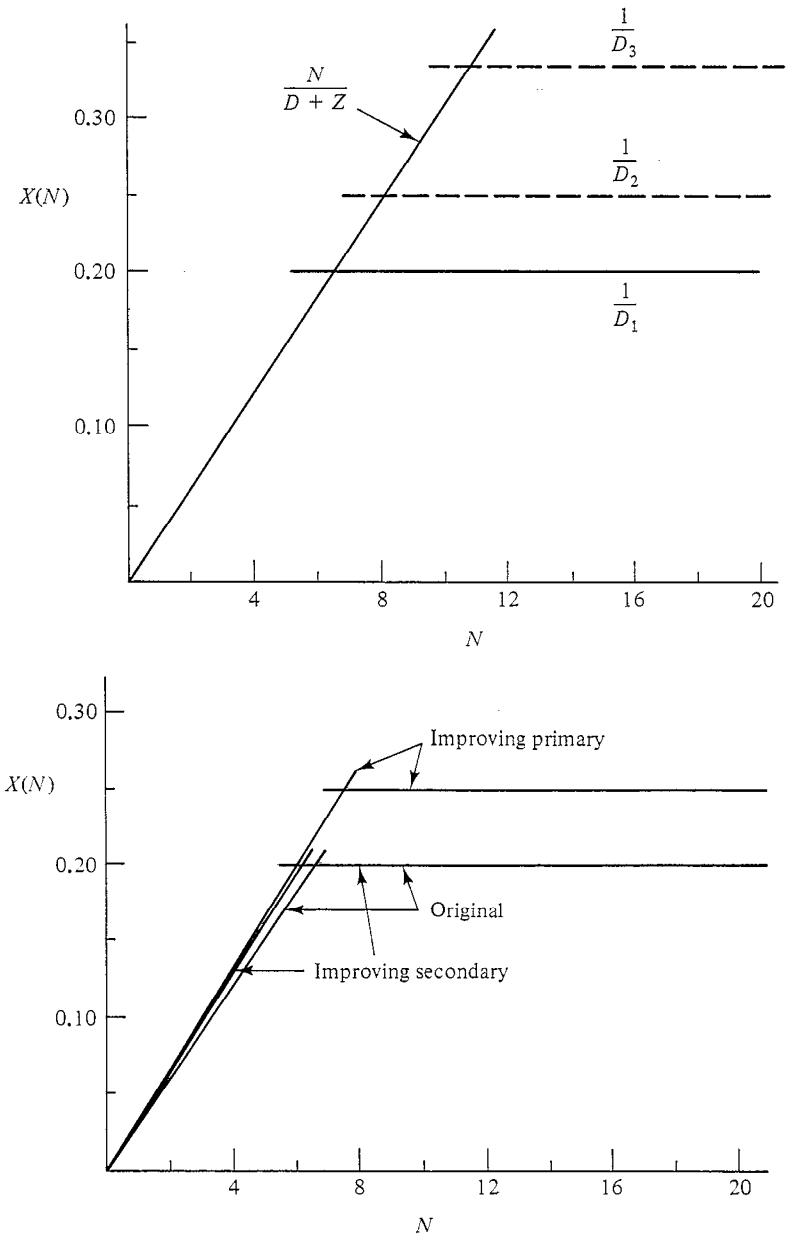

Figure 3: Optimistic (i.e., upper) throughput asymptotic bounds.

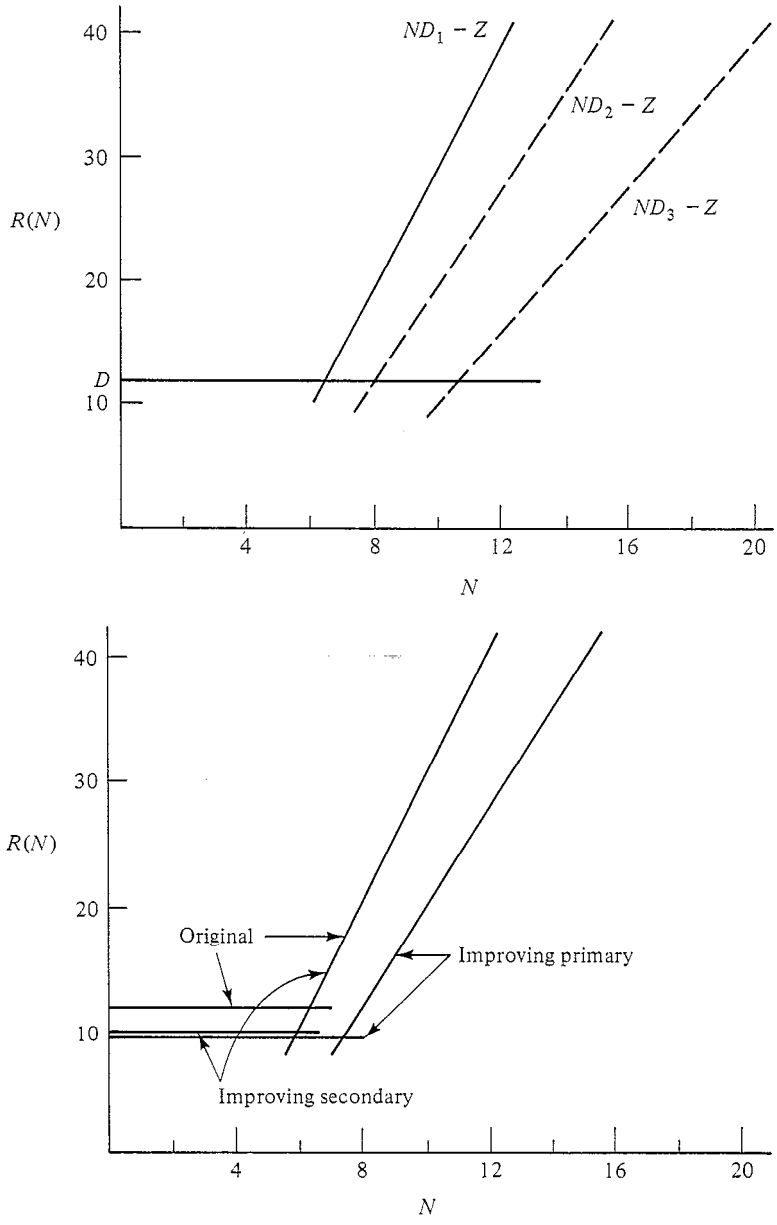

Figure 4: Response time asymptotic bounds.

Exercise 4

The `http://perflib.net` website is deployed onto a machine that runs one instance of the Nginx[3] web server and two parallel instances of the PostgreSQL[4] database server (each on a separate disk).

In our closed, terminal workload environment, we simulated the activity of N clients with 15 seconds of think time and we measured the busy time and the number of jobs completed at each center (except for the web server) for a time interval of 900 seconds. During this time interval the system rendered 200 HTTP responses back to the clients.

The measurements were as follows:

- the web server has a busy time of 400 seconds,
- the first database server has a busy time of 100 seconds, and completed $2,000$ operations,
- the second database server has a busy time of 600 seconds, and completed $20,000$ operations.

1. Draw the queuing network model of the deployment.
2. Can you spot the bottleneck?
3. How many visits, to each database server, are required per HTTP request?
4. How much service time does a visit take on each database server?
5. What is the effect of replacing the Nginx web server with the Cherokee[5] web server, which is twice as fast?
6. The database administrator claims that the load between the two database servers is perfectly balanced. Is he correct? Why?
7. If you are given enough money to buy a new disk to run an additional database server, how would you modify the system?
8. What would be a non-obtrusive modification (i.e., involving no costs at all) to alleviate the bottleneck?
9. Calculate the number of visits to the web server taken by each HTTP request in the original configuration, but with the Cherokee web server instead of Nginx.

Answer of exercise 4

1. The diagram can be generated with the JMTGraph[6] tool, for example. However, it is quite irrelevant because we already know all the D_k, thus the layout of the network does not affect our calculations.

[3] `http://nginx.net/`
[4] `http://www.postgresql.org/`
[5] `http://www.cherokee-project.com`
[6] `http://jmt.sf.net`

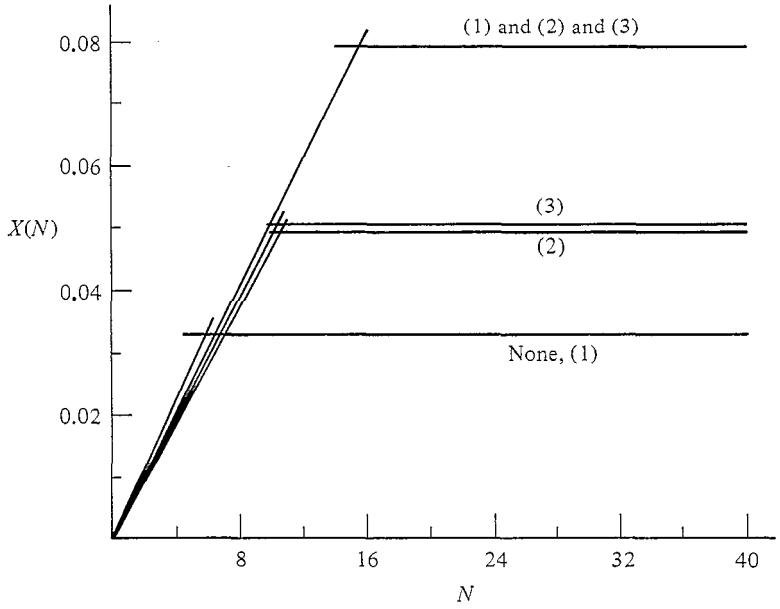

Figure 5: Asymptotic bounds for the throughput.

2. To find the bottleneck, the D_{max} is needed and so all the D_k. These are simply the time each center is busy *for each job* (note, *not* for each visit). The total number of completed jobs is known. Then:

$$D_k = \frac{D_k}{C}$$

$$D_{WS} = \frac{B_{WS}}{C} = \frac{400s}{200} = 2.0s$$

$$D_{DB1} = \frac{B_{AS}}{C} = \frac{100s}{200} = 0.5s$$

$$D_{DB2} = \frac{B_{DB}}{C} = \frac{600s}{200} = 3.0s$$

   At this point we can say that $D_{max} = \max_k D_k = D_{DB2}$ thus the slower database server is the first resource that cause the whole system's saturation.

3. The number of visits on each center is the number of times *for each job* (1 HTTP request = 1 job) that a center is demanded. Thus, similarly to what we have just done to calculate the service demand:

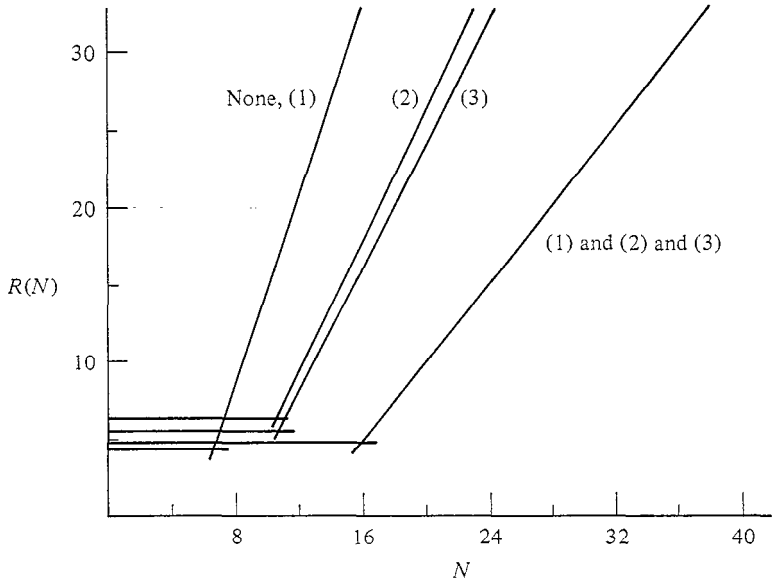

Figure 6: Asymptotic bounds for the response time.

$$V_k = \frac{C_k}{C}$$

$$V_{DB1} = \frac{C_{DB1}}{C} = \frac{20,000}{200} = 100$$

$$V_{DB2} = \frac{C_{DB2}}{C} = \frac{20,000}{200} = 100$$

4. Now that we have both V_{DB1} and D_{DB1} (along with V_{DB2} and D_{DB2}) we can resort to the definition of service demand to calculate the service time:

$$D_k = S_k V_k \Leftrightarrow S_k = \frac{D_k}{V_k}$$

Recall that the service demand considers the number of visits whereas the service time only consider the time spent to serve one visit.

For the database servers we have:

$$S_{DB1} = \frac{D_{DB1}}{V_{DB1}} = \frac{0.5s}{10} = 0.05ms$$

$$S_{DB2} = \frac{D_{DB2}}{V_{DB2}} = \frac{3.0s}{100} = 0.03ms$$

Note that the slower database takes less time on each visit with respect to the faster one. However, the former is demanded for less time on each job while the latter is demanded six times more.

5. A faster web server means a decreased service demand. Since the speed is doubled we have a factor of 2. Thus:

$$D'_{WS} = \frac{D_{WS}}{2} = \frac{2.0s}{2} = 1.0s$$

Since the slow database is the bottleneck, it limits the throughput to:

$$X(N) \geq \frac{1}{D_{DB2}} = \frac{1}{3.0s} = 0.333s$$

Unfortunately, the maximum throughput remains the same even though we switch to a faster web server (see the asymptotic bound analysis in Figure 5).

6. As we have seen, one database is demanded $D_{DB1} = 0.5s$ for each job, while the other one is demanded $D_{DB2} = 3.0s$ for each job. This means that the slower one is busy six times more than the faster one. Therefore, they are by no means balanced.

7. We may attempt to alleviate the bottleneck by putting an additional database to balance the slower one. The optimal way is to split D_{DB2} equally across the two databases:

$$D'_{DB2} = \frac{D_{DB2}}{2} = 1.5s$$

$$D_{DB3} = \frac{D_{DB2}}{2} = 1.5s$$

We have assume that $S_{DB2} = S_{DB3} = 0.03s$.

8. If no new hardware is allowed, the only option is to alleviate the bottleneck by re-balancing the load across the two existing databases. This means that the database administrator would have to shift some files from the slower database to the faster one.

To calculate the new service demand for each database, we have to assume that the service time *must* remain the same, being it an intrinsic characteristic of the database server. Thus:

$$\begin{aligned} V_{DB1} + V_{DB2} &= V_{DB1} + V_{DB2} \\ V_{DB1} + V_{DB2} &= 110 \end{aligned}$$

let's now divide and multiply for the same quantity, the service time:

$$\frac{V_{DB1} S_{DB1}}{S_{DB1}} + \frac{V_{DB2} S_{DB2}}{S_{DB2}} = 110$$

we notice that $D_k = V_k S_k$, thus:

$$\frac{D_{DB1}}{S_{DB1}} + \frac{D_{DB2}}{S_{DB2}} = 110$$

but we want that:

$$D_{DB} = D_{DB1} = D_{DB2}$$

thus:

$$\begin{aligned} D_{DB}\left(\frac{1}{S_{DB1}} + \frac{1}{S_{DB2}}\right) &= 110 \\ D_{DB} &= 110 \cdot (S_{DB1} + S_{DB2}) \\ D_{DB} &= \frac{110}{53.333\ 1/s} = 2.063s \end{aligned}$$

And this pushes the bottleneck up to $X(N) \geq 0.384 r/s$. The new visit counts due to the new service demands on each database are:

$$V_{DB1} = \frac{D_{DB}}{S_{DB1}} = \frac{2.063s}{0.05s} = 41$$

$$V_{DB2} = \frac{D_{DB}}{S_{DB2}} = \frac{2.063s}{0.03s} = 69$$

We can combine all the changes together: the Cherokee web server ensures $D'_{WS} = 1.0s$ and $D_{DB} = D_{DB1} = D_{DB2} = D_{DB3}$. The total number of visits and the service time of each database remains the same, thus:

$$
\begin{aligned}
D_{DB}\left(\frac{1}{S_{DB1}}+\frac{1}{S_{DB2}}+\frac{1}{S_{DB3}}\right) &= 110 \\
D_{DB}\left(\frac{1}{0.05s}+\frac{1}{0.03s}+\frac{1}{0.03s}\right) &= 110 \\
D_{DB} &= \frac{110}{86.667\ 1/s} = 1.269s
\end{aligned}
$$

9. This can be calculated by summing the total number of visits $V_{DB1} + V_{DB2}$, plus one extra visit to the web server needed to respond to the client. This makes $V_{WS} = 111$ in the original configuration.

   On this we can calculate the service time of the web server, that is $S_{WS} = \frac{2.0s}{111} = 180ms$.