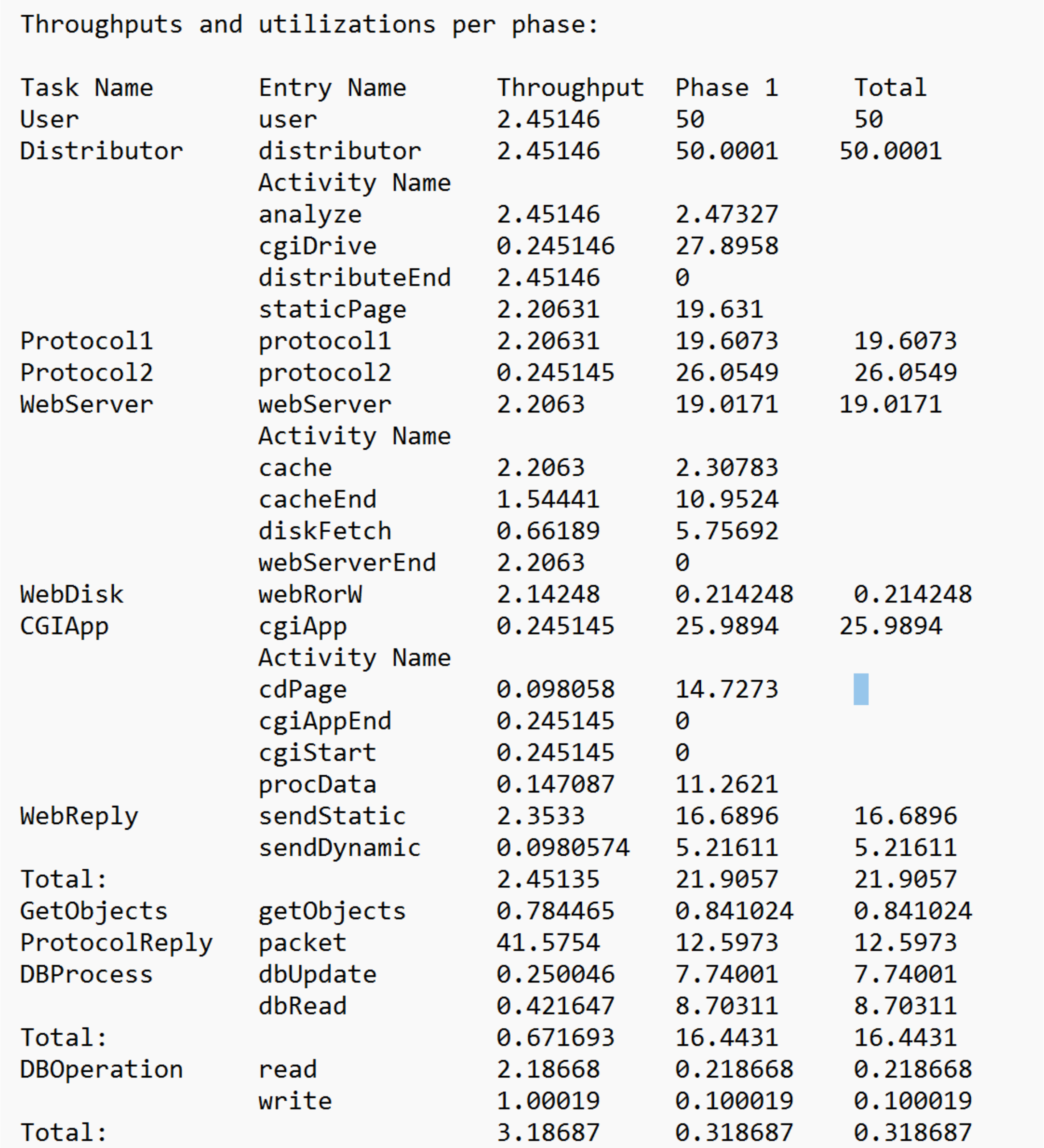
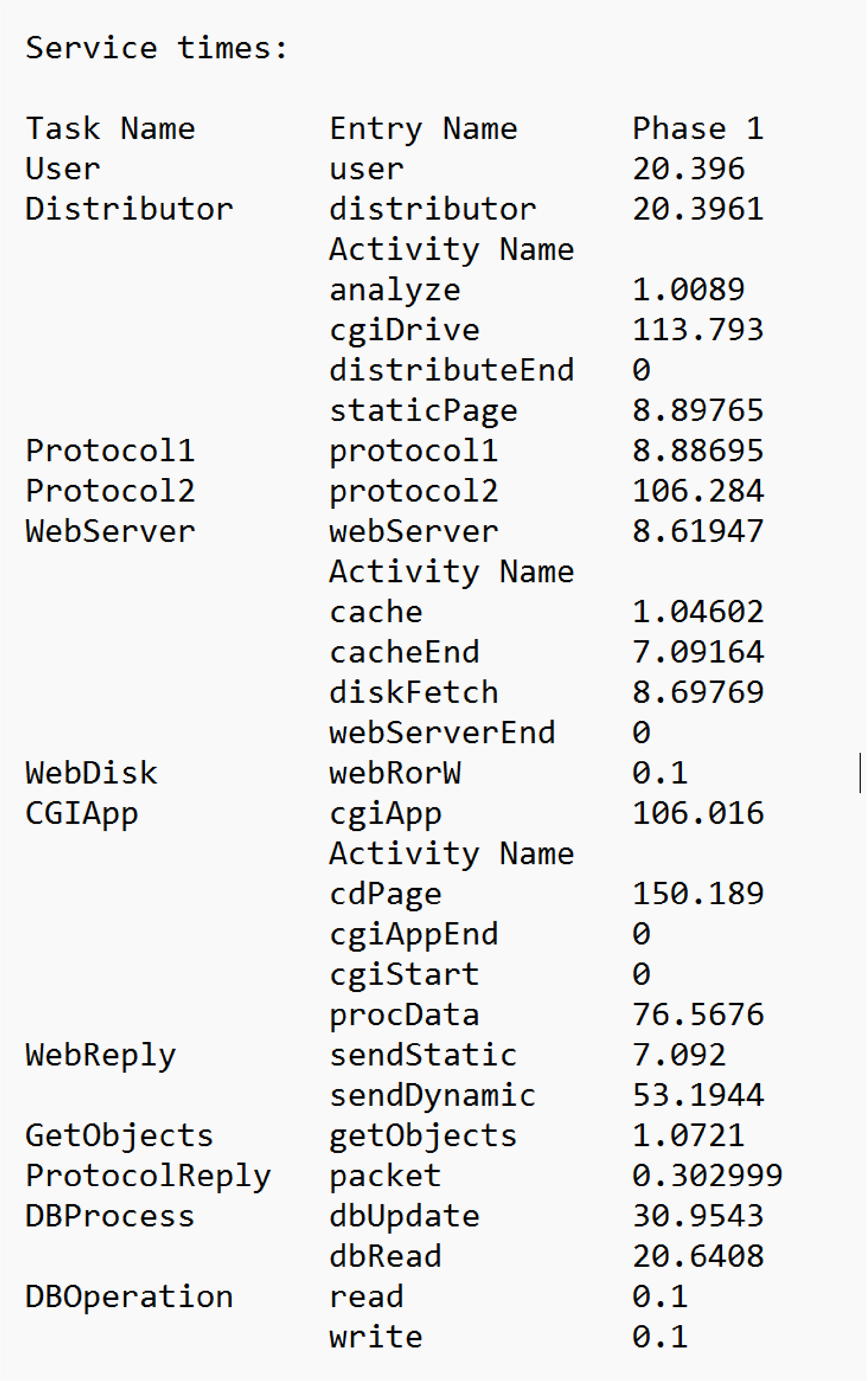
3. The final thing I will do is add a Distributor to my project.

The Distributor task includes operations such as analyze and distributeEnd. It leverages the analyze activity to send work to protocol 1 for static pages (staticPage) or protocol 2 for CGI requests (cgiDrive). This enables the distributor to properly manage and route incoming work according to the type of request.
The distributor balances the workload by dividing jobs between protocols 1 and 2 based on the kind of request (static or CGI). This efficiently decreases the resource demands on each protocol while keeping to the Centering Principle which focuses on the software components with the greatest impact on performance by releasing dominating workload functions.
This output file prints out the results after adding the distributor.



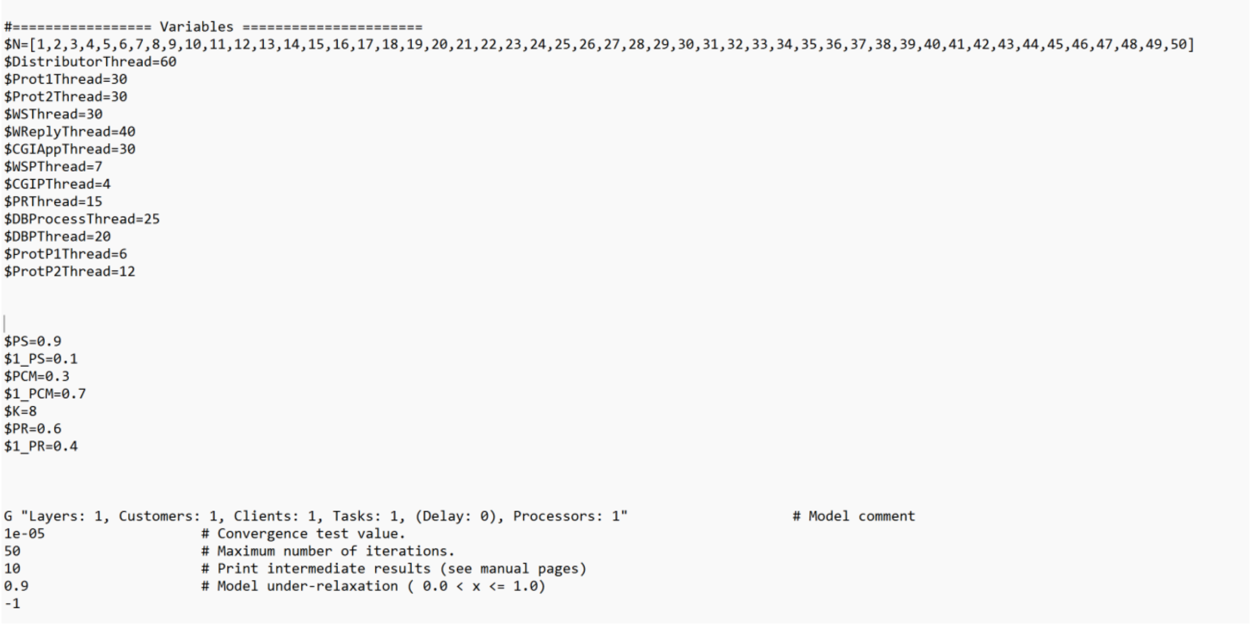
```
Throughputs and utilizations per phase:

Task Name       Entry Name      Throughput  Phase 1     Total
User            user            2.45146     50          50
Distributor     distributor     2.45146     50.0001    50.0001
                Activity Name
                analyze         2.45146     2.47327
                cgiDrive        0.245146    27.8958
                distributeEnd   2.45146     0
                staticPage      2.20631     19.631
Protocol1       protocol1       2.20631     19.6073     19.6073
Protocol2       protocol2       0.245145    26.0549     26.0549
WebServer       webServer       2.2063      19.0171    19.0171
                Activity Name
                cache           2.2063      2.30783
                cacheEnd        1.54441     10.9524
                diskFetch       0.66189     5.75692
                webServerEnd    2.2063      0
WebDisk         webRorW         2.14248     0.214248    0.214248
CGIApp          cgiApp          0.245145    25.9894    25.9894
                Activity Name
                cdPage          0.098058    14.7273
                cgiAppEnd       0.245145    0
                cgiStart        0.245145    0
                procData        0.147087    11.2621
WebReply        sendStatic      2.3533      16.6896     16.6896
                sendDynamic     0.0980574   5.21611     5.21611
Total:                          2.45135     21.9057     21.9057
GetObjects      getObjects      0.784465    0.841024    0.841024
ProtocolReply   packet          41.5754     12.5973     12.5973
DBProcess       dbUpdate        0.250046    7.74001     7.74001
                dbRead          0.421647    8.70311     8.70311
Total:                          0.671693    16.4431     16.4431
DBOperation     read            2.18668     0.218668    0.218668
                write           1.00019     0.100019    0.100019
Total:                          3.18687     0.318687    0.318687
```

```
Entry execution demands:

Task Name       Entry Name      Phase 1
User            user            0
Distributor     Activity Name
                analyze         1
                cgiDrive        0
                distributeEnd   0
                staticPage      0
Protocol1       protocol1       0.25
Protocol2       protocol2       0.25
WebServer       Activity Name
                cache           1
                cacheEnd        0
                diskFetch       1
                webServerEnd    0
WebDisk         webRorW         0.1
CGIApp          Activity Name
                cdPage          5
                cgiAppEnd       0
                cgiStart        0
                procData        15
WebReply        sendStatic      0.8
                sendDynamic     0.8
GetObjects      getObjects      1
ProtocolReply   packet          0.25
DBProcess       dbUpdate        30
                dbRead          20
DBOperation     read            0.1
                write           0.1
```

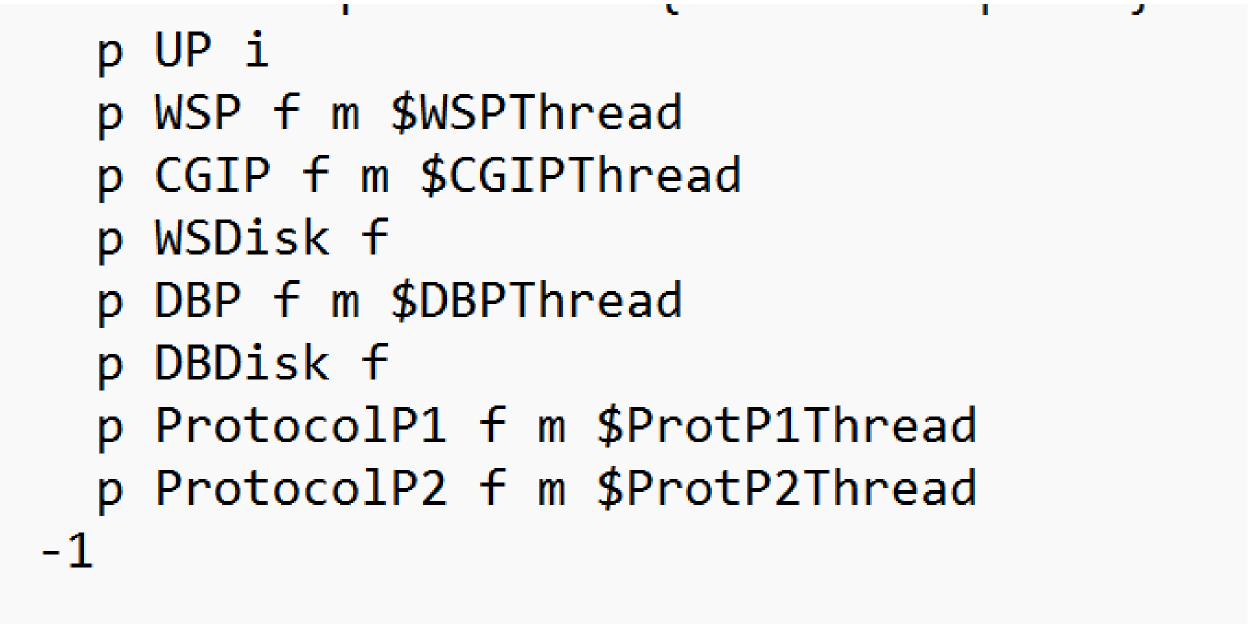


```
Service times:

Task Name       Entry Name      Phase 1
User            user            20.396
Distributor     distributor     20.3961
                Activity Name
                analyze         1.0089
                cgiDrive        113.793
                distributeEnd   0
                staticPage      8.89765
Protocol1       protocol1       8.88695
Protocol2       protocol2       106.284
WebServer       webServer       8.61947
                Activity Name
                cache           1.04602
                cacheEnd        7.09164
                diskFetch       8.69769
                webServerEnd    0
WebDisk         webRorW         0.1
CGIApp          cgiApp          106.016
                Activity Name
                cdPage          150.189
                cgiAppEnd       0
                cgiStart        0
                procData        76.5676
WebReply        sendStatic      7.092
                sendDynamic     53.1944
GetObjects      getObjects      1.0721
ProtocolReply   packet          0.302999
DBProcess       dbUpdate        30.9543
                dbRead          20.6408
DBOperation     read            0.1
                write           0.1
```

SNIIPPETS OF THE CODE WILL BE DISPLAYED TO EXPLAIN HOW PERFORMANCE PATTERNS AND PRINCIPLES WILL BE APPLIED.



```
#================= Variables ======================
$N=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50]
$DistributorThread=60
$Prot1Thread=30
$Prot2Thread=30
$WSThread=30
$WReplyThread=40
$CGIAppThread=30
$WSPThread=7
$CGIPThread=4
$PRThread=15
$DBProcessThread=25
$DBPThread=20
$ProtP1Thread=6
$ProtP2Thread=12


$PS=0.9
$1_PS=0.1
$PCM=0.3
$1_PCM=0.7
$K=8
$PR=0.6
$1_PR=0.4


G "Layers: 1, Customers: 1, Clients: 1, Tasks: 1, (Delay: 0), Processors: 1"                    # Model comment
1e-05                   # Convergence test value.
50                      # Maximum number of iterations.
10                      # Print intermediate results (see manual pages)
0.9                     # Model under-relaxation ( 0.0 < x <= 1.0)
-1
```



```
  p UP i
  p WSP f m $WSPThread
  p CGIP f m $CGIPThread
  p WSDisk f
  p DBP f m $DBPThread
  p DBDisk f
  p ProtocolP1 f m $ProtP1Thread
  p ProtocolP2 f m $ProtP2Thread
 -1
```

```
#================= Tasks ======================
T 0
# SYNTAX: t TaskName TaskType EntryList -1 ProcessorName [flags]
#   TaskName is any string, globally unique among tasks.
#   TaskType = r {reference or user task}
#            | n {other}
#   flags = m <int> {multithreaded}
#         | i {infinite or delay server}
#         | z <real> {think time}
#         | <int> {task priority}
  t User r user -1 UP m $N
  t Distributor n distributor -1 ProtocolP1 m $DistributorThread
  t Protocol1 n protocol1 -1 ProtocolP1 m $Prot1Thread %f $thruProtocol %u $uProtocol
  t Protocol2 n protocol2 -1 ProtocolP1 m $Prot2Thread
  t WebServer n webServer -1 WSP m $WSThread %f $thruWebServer %u $uWebServer
  t WebDisk n webRorW -1 WSDisk %f $thruWDisk %u $uWDisk
  t CGIApp n cgiApp -1 CGIP m $CGIAppThread %f $thruCGIAPP %u $uCGIAPP
  t WebReply n sendStatic sendDynamic -1 WSP m $WReplyThread %f $thruWebReply %u $uWebReply
  t GetObjects n getObjects -1 WSP %f $thruGetObjects %u $uGetObjects
  t ProtocolReply n packet -1 ProtocolP2 m $PRThread %f $thruProtReply %u $uProtReplly
  t DBProcess n dbUpdate dbRead -1 DBP m $DBProcessThread %f $thruDBProcess %u $uDBProcess
  t DBOperation n read write -1 DBDisk %f $thruDBOpt %u $uDBOpt
-1
```

```
#================= Entries =====================
E 0
# SYNTAX-FORM-A: Token EntryName Value1 [Value2] [Value3] -1
#   EntryName is a string, globally unique over all entries
#   Values are for phase 1, 2 and 3 {phase 1 is before the reply}
#   Token indicate the significance of the Value:
#       s - HostServiceDemand for EntryName
#       c - HostServiceCoefficientofVariation
#       f - PhaseTypeFlag
# SYNTAX-FORM-B: Token FromEntry ToEntry Value1 [Value2] [Value3] -1
#   Token indicate the Value Definitions:
#       y - SynchronousCalls {no. of rendezvous}
#       F - ProbForwarding {forward to ToEntry rather than replying}
#       z - AsynchronousCalls {no. of send-no-reply messages}
# ---------- user ----------
  s user 0 -1
  y user distributor 1 -1 %w1 $wUser
# ---------- distributor  ----------
  A distributor analyze
# ---------- protocol1 ----------
  s protocol1 0.25 -1
  y protocol1 webServer 1 -1 %w1 $wProtocol1
# ---------- protocol2 ----------
  s protocol2 0.25 -1
  y protocol2 cgiApp 1 -1 %w1 $wProtocol2
# ---------- webserver ----------
  A webServer cache
# ---------- webDisk ----------
  s webRorW 0.1 -1
# ---------- sendPage ----------
  s sendStatic 0.8 -1
  y sendStatic packet 16 -1 %w1 $wSendStatic
  s sendDynamic 0.8 -1
  y sendDynamic getObjects 8 -1
  y sendDynamic packet 40 -1
  s getObjects 1 -1
  y getObjects webRorW 0.2 -1
# ---------- protocol Reply ----------
  s packet 0.25 -1
# ---------- CGI Application ----------
  A cgiApp cgiStart
# ---------- CGI Application Database ----------
  s dbUpdate 30 -1
  y dbUpdate read 2 -1
  y dbUpdate write 4 -1
  s dbRead 20 -1
  y dbRead read 4 -1
  s read 0.1 -1
  s write 0.1 -1
-1
```

```
#================= Activities ======================
A Distributor
        s analyze 1
        s staticPage 0
        y staticPage protocol1 1
        s cgiDrive 0
        y cgiDrive protocol2 1
        s distributeEnd 0
:
        analyze -> (0.9)staticPage + (0.1)cgiDrive;
        staticPage + cgiDrive -> distributeEnd;
        distributeEnd[distributor]
-1

A WebServer
        s cache 1
        s diskFetch 1
        y diskFetch webRorW 3
        y diskFetch sendStatic 1
        s cacheEnd 0
        y cacheEnd sendStatic 1
        s webServerEnd 0
:
        cache -> (0.3)diskFetch + (0.7)cacheEnd;
        cacheEnd + diskFetch -> webServerEnd;
        webServerEnd[webServer]
-1

A CGIApp
        s cgiStart 0
        s procData 15
        y procData dbUpdate 1.7
        y procData sendStatic 1
        s cdPage 5
        y cdPage dbRead 4.3
        y cdPage sendDynamic 1
        s cgiAppEnd 0
        #s cgiEnd 0
        #y cgiEnd sendStatic 1
:
        cgiStart -> (0.6)procData + (0.4)cdPage;
        procData + cdPage -> cgiAppEnd;
        cgiAppEnd[cgiApp]
-1
```

PERFORMANCE PRINCIPLES

My new model incorporates these new principles and so I believe can be applied to the original model.

1. Centering Principle:
   The Distributor task in LQN code distributes work to protocol tasks (Protocol1 and Protocol2) and the WebServer. This emphasis on the Distributor task as a dominating workload function exemplifies the centering concept, which involves improving system performance by focusing on the most important tasks.

2. Locality Principle: The LQN code prioritizes keeping actions and functions near to the resources they use. For example, the WebServer and CGIApp tasks contact other tasks (such as webRorW and dbUpdate) for data processing and retrieval, ensuring that the activities are local to the resources being used.

3. Parallel Processing Principle: The code uses many threads (e.g., m $DistributorThread, m $Prot1Thread, m $Prot2Thread) to handle requests simultaneously. This can speed computer processing by dividing work into multiple concurrent processes.

4. Spread the Load Principle
   The LQN code distributes workload among jobs and processors, such as between Protocol1 and Protocol2. This can help to balance the workload and prevent contention delays.

Performance patterns:

Batching:

The model includes tasks for batching requests, such as dbUpdate, which combines numerous reads and writes into batch operations. This is consistent with the "Processing Versus Frequency" theory and can increase efficiency.

Flex Time: The model distributes tasks across processors (p UP i, p WSP f m $WSPThread, p DBP f m $DBPThread) to manage load over time.

Locality: Tasks are aligned with their most frequent uses, such as webServer, webRorW, and sendStatic. This demonstrates a focus on locality and efficient processing.

Performance Antipatterns:

"God" Class: The Distributor and User jobs handle a substantial percentage of the work, which can lead to excessive traffic and decreased performance if they become highly centralized.

Circuitous Treasure Hunt:

The procData activity in CGIApp interacts with several activities, including dbUpdate, sendStatic, and dbRead, which may require many search operations to obtain data.

One-Lane Bridge: Some tasks run on single processors (e.g., p DBP f m $DBPThread), which might cause bottlenecks if only one process can run at a time.

Traffic Jam:

Inadequate management of task interactions (e.g., User, Distributor, Protocol1, Protocol2, and WebServer) can result in backlogs and unpredictable response times.

CONCLUSION:

In conclusion, this layered queuing network model of a web server system helps us understand how various tasks entries and activities have a major impact in our system. To ensure we are designing a high performance software continuous monitoring and evaluation is needed to spot out any errors or bottlenecks can damage the software.