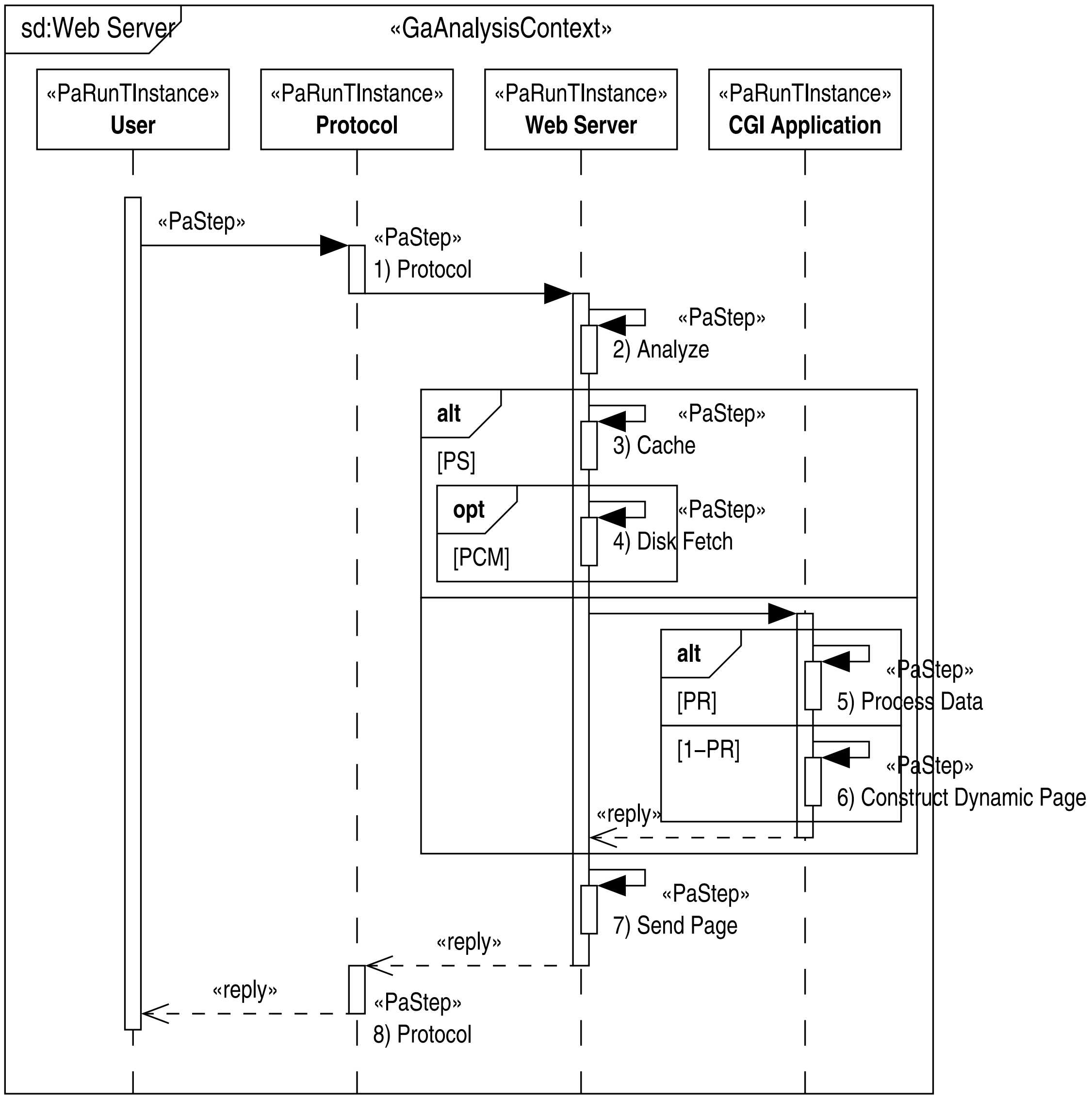
Carleton University
Department of Systems and Computer Engineering
SYSC5101, Design of High Performance Software, Winter 2020
Project 1

This document describes the first project template based on LQN. The student groups that select this template should start by changing the original description of the web server system as follows: modify the given scenario, add one or two new scenarios, modify/add new tasks, entries and hardware resources. This will change the details of the corresponding LQN model. The steps of the "What to do section" should be followed as they are described or changed. You are encouraged to do as many modifications as possible, as more modifications will make for a better project.

Model template description

Consider a basic web server system. The main scenario whose performance we would like to evaluate is described by the following sequence diagram.



The User process runs on a PC and represents a population of users. The Protocol Stack, Web Server and Application processes run on the Web Server Processor, WSP. Static pages are stored on a disk WSDisk. The Application invokes a database server which runs on a Database Processor DBP, which uses its own disk DBDisk.

The operations identified by «PaStep» are described below, with their host demands in msec:

1. Protocol operations (0.25 ms per packet; a request has 1 packet)
2. Analyze the HTTP request to determine its type (1 ms per request)

First type: request a static page, with probability PS

3. Retrieve page from cache if present (1 ms)
4. Retrieve page from disk if not in cache (with probability *PCM* of cache miss, demand 1 ms)

Second type: CGI request with user data, prob. PR*(1.0 – PS), where PR=0.6

5. Process the user data in the application, store it in the database, determine the next page to be sent.

Third type: CGI request without data, probability (1-PR)*(1.0 – PS)

6. Construct dynamic page
7. Send page, possibly with embedded objects
8. Protocol operations (0.25 ms per packet; a static page has 16 packets, an embedded graphics object takes 3 packets)

Operations 4, 5, 6 and 7 contain embedded operations as follows:

4. Retrieve: 1 ms host demand, and one static web page retrieve from disk WSDisk (operation 9)
5. Process user data: 15 ms host demand, 1.7 database update operations (operation 11) to store user data
6. Construct dynamic page: 5 ms host demand, and 4.3 database read operations (operation 12)
7. Send page: sends the HTML page (operation 13) and an average of K embedded graphic objects (operation 14).

The embedded operations have the following demands:

9. Static web page retrieve: average of 3 disk block read operations on WSP/WSDisk (operation 10)
10. Disk block read or write: 0.1 ms host demand, one disk read on attached disk device
11. Database update: 30 ms CPU (on host DBP), 2 disk reads and 4 disk write operations on *DBP/DBDisk* (operation 10, on *DBP/DBDisk*)

12. Database read: 20 ms CPU (on host *DBP*), 4 disk read operations (operation 10, on *DBP/DBDisk*)
13. Send static page: 0.8 ms host (*WSP*) for the HTML page.
14. Retrieve and send embedded object: 1 ms host, 0.2 average operations on WSDisk (operation 10).

Disk service times are 10 msec in both cases, and both are single disks; the processors are single processors. Remember that we assume there is no limit to the number of users that can be active in the software components at one time, but the disks and processors are single servers. The scenario is executed sequentially for each request. We will ignore the internet delay and the internal communications delays between *WSP* and *DBP*.

We will use the above web server scenario with a closed workload of *NU* Users, each with a "think time" of *ZU* minutes between requests for web pages; note that each User has its own PC. The tasks "ProtocolStack", "WebServer" and "CGIApplication" are all running on processor WSP, and the task DBServer on DBP.

What to do:

(1) Build an LQN model with a User task for each user and browser together, make all tasks single threaded, *PS* = probability of a static page request = 0.9, *PCM* = prob. of cache miss = 0.3, and K = 8 embedded objects per page, on average. Use *processor sharing* scheduling at the processors.

- Build the LQN model.
- Increase the number of Users, *NU,* until reaching saturation (i.e., levelling off of throughput).
- What resources are saturated in the LQN? Identify the bottleneck.
- How sensitive is the result to the cache hit rate PCM? In order to find out, solve the model for different PCM values and compare the system performance obtained for different PCM. High sensitivity means that the results are substantially affected, low sensitivity means low impact.

(2) Depending on the type of bottleneck - software or hardware - apply changes to the LQN model that will alleviate the bottleneck.

- Solve the model for increasing number of users until reaching saturation. Display the system throughput and response time and compare it with the results from (1). Discuss the performance improvement.
- Apply changes to the model to alleviate the current bottleneck and plot the new throughput and response time on the same graph as the previous results. Repeat the improvement procedure until the performance improvement is considered satisfactory.

(3) Investigate what Software Performance Engineering principles (to be presented in lectures) can be successfully applied to the original model.

Project Marking Scheme [Total 35]

a) [5] Variation from Template - the more changes, the better.

b) [5] Software Model specification (text, diagrams)

c) [5] Performance Model - construction, explanation how it relates to the software model and how the performance parameters (e.g., service demands, numbers of calls) have been obtained.

d) [10] Performance Analysis (covers perf analysis throughout the project):

- LQN: clear bottleneck identification and mitigation, alternatives, sensitivity analysis, presentation of results
- SPN: Exec. graph analysis + QN analysis, sensitivity analysis

e) [10] Improvements (principles, patterns, antipatterns) and their application.