***Chapter 4***

**SIMULATION AND RESULTS**

**4.1 Introduction to discrete event simulation**

Simulation can be defined as the experimentation with a model in order to obtain information about the dynamic behaviour of a system. Instead of experimenting with the system, the experiments are performed with a model of the system. Simulation is mainly used when experimenting with the real system is too expensive, too dangerous, or not possible**.**

There are three categories of simulation models, defined by the way the system state changes:

**Continuous** In this, the state varies with time continuously. These kind of systems are typically described by sets of differential equations.

**Discrete** Here, the state changes only at discrete intervals of time (event times).

**Combined continuous and discrete** In thisthe state changes instantaneously at event times i.e, in between consecutive event times the system state may vary continuously.

There are basically three approaches that can be used for discrete event simulation:

1. Event based 2. Activity based 3. Process based.

In this report we will only consider so-called discrete event simulation. In discrete event simulation the model is discrete, and the simulated clock always jumps from one event time to the most imminent event time. At each event time the corresponding action is performed, and simulated time is advanced to the next time when some action is to occur. Thus, discrete event simulation assumes that nothing happens between successive state changes.

**The event-based approach**

In the event-based approach the model consists of a collection of events. Each event models a state change and is responsible for scheduling other events that depends on that event.

Each event has associated an event time and some actions to be executed when the event occurs. Event-based simulation is the simplest and most common implementation style of discrete event simulation because it can be implemented in any programming language.

**The activity-based approach**

In the activity-based approach the model consists of a collection of activities. Each activity models some time-consuming action performed by an entity. Each activity has associated a starting condition, some actions to be executed when the activity starts, the duration of the activity, and some actions to be executed when the activity finishes. Whilst the activity approach is relatively easy to understand, it normally suffers from poor execution efficiency compared to the event-based approach.

**The process-based approach**

In the process-based approach the model consists of a collection of processes. Each process models the life cycle of an entity and is a sequence of logically related activities ordered in time. Since processes resemble objects in the real world, process-based simulation is often easy to understand. Implementation, however, is not easy and execution efficiency may be poor if the implementation is not done properly.

In this thesis, we used discrete event simulation to do experimentation on the proposed approach.

**4.2. Simulation Assumptions**

In this, we assume the following assumptions to do simulation.

1. Every web object consists 2 fields: web page id and time
2. Each web object is of same size
3. Object life time is in seconds
4. All delays are negligible

**Metric using in the simulation is hit ratio only:** Since we are assuming all objects are of same size we can’t use byte hit ratio or byte precision as a metric. The only possible metric is hit ratio for web caching, precision for prefetching.

**4.3. Events present in the simulation**

**CLIENT THREAD** continuously sends page requests and these requests will stored in PAGE REQUEST EVENTS QUEUE.

**WEB CACHE data structure** all web objects and their corresponding lifetimes are cached in to this data structure.

**WEB CACHING THREAD** It processes each request present in the PAGE REQUEST EVENTS QUEUE and counts the hit ratio if the object is present in WEB CACHE data structure and if it is fresh. Freshness of object is checked by using object life time.

**PREFETCHING THREAD** While WEB CACHING THREAD is processing the request

PREFETCHING THREAD fetches the next sub sequent user’s request based on the past history.

**4.4. MODULES CREATED IN JAVA**

1. **PAGE REQUEST EVENT** In this module, every objects id and corresponding time in milliseconds is generated.
2. **CLIENT** Here, for every client page requests are created with the help of PAGE REQUEST EVET module and those are kept in a QUEUE.
3. **WEB CACHE** This module is for caching the objects. If the size of the WEB CACHE is full according to the replacement policy an object is evicted in order to create space for the arrival object.
4. **WEB CACHING** In this module, page requests present in the QUEUE are processed with the help of WEB CACHE module. If the object is present in it and it is not staled one hit ratio count is increased.
5. **TRAIN SET** This module is for capturing the users past web sessions.
6. **PREDICTION** Here, with the help of TRAIN SET the prediction engine runs and predicts the next user’s request.
7. **PREFETCHING** This module fetches the next user request and stores in the WEB CACHE module.

**4.5. Results:**