Chapter-3

Integrated-cost-Model

In this affiliate we tend to outline the ambit of the apriorism by presenting an additional meticulous explanation of the system model planned within the apriorism. Furthermore, this section present an integrated cost model established on the system design. There are two facilities provided by the cost-model: it contributes a certain mode of shaping issues examined in advanced affiliate; & it catches the recital repercussions of the simultaneous interactions among the various buffer-management methods and also the client threads.

The affiliate begins by illustrating the system & allusions-models utilized during the affiliate. Exploiting these models, we ascertain an integrated-cost-model portraying in what way the four buffer-management regions have an effect on system execution. Finally, we tend to define additional sensible allusions models that are utilized in the affiliate.

3.1 System-Model

Here we model multiple-user database which permits contemporaneous accomplishment of user requests. The model has the subsequent system entities (as shown in segment2.4 for a consultation of probable system designs):

1. The server of the database attending one or additional client procedures exploiting one or additional CPUs.
2. The pages are pre-loaded from the diskette into cache by a pre-fetch thread.
3. A diskette exclusively available to the server.
4. A dynamic agglomeration thread that sporadically modifies the article to page-loading.
5. A diskette queue that reorganizes disk page demands.
6. A main memory page cache that the server & clients claim.
7. Variety of concomitantly executing processes of client on identical system because the server, executing application’s demanding object’s from the server.

Fig3.1 displays a figure of the system-model discovered. We select to concentrate our devotion on one standalone node of a peer to peer machine. The intention, as defined in segment2.4, is that we'd prefer to concentrate our study on the results of diskette I/O.
The requests of the clients are object requests in this model and these requests are converted to page requests by object to page mapping at the server. The request send by the client can be jammed till the demanded object turn out to be accessible. The clients & server have the common main-memory page cache. The maximum K pages can be hold by the cache. It's instantly came to the demanding thread or process; if a demanded page is found within the cache; otherwise, the disk-block mapping is used to mapped the requested page on a disk-block & place it on the server-queue. The mapping from page to disk-block permits the server to find the page on the diskette which is requested. The rearrangement of disk-page requests are done by server-queue in such a manner that depends on the client scheduling program & the diskette. At the last, the requests which are now rearranged & emerging from the queue are directed to the diskette. So as to stay the cost-models easy we are not modelling the results that the page to disk-block mapping has on performance.

![Diagram](image)

**Fi 3.1**: Illustration of system-model. Ellipse objects are concomitantly executing threads or processes. Every thread & process is accomplished on an equivalent system.

A fetch will action as an outcomes of a cache error or as an outcomes of a predicted failure (prefetch). Pre-fetch threads are used to start pre-fetches. On 1 time only 1 fetch is permitted, i.e., there's not any concurrency on the diskette I/O level. Concurrency is also not allowed among read & write diskette I/O. Once a page fetch is demanded & if the cache is occupied, then the page which is in the cache should be removed. If the removed page is fresh, it's rejected;
otherwise it's drafted back to diskette before the demanded page is laden. The development of the fetch means that no accesses are possible for any incoming page & any evicted page. All the requests are placed on a server queue like diskette write-requirements, read-requirements. The alteration from object to page-mapping is done by dynamic-clustering thread that periodically breaks the programs execution & reorganizes the DB. One page is mapped by accurately one object.

3.2 Allusion-Model
Here we justify the allusion model utilized by the cost-models of this apriorism. A vital quality of the allusion model is that it permits the meaning of numerous coincident transactions. Here, the allusion-model is simply utilized reflect sequence’s of client-thread demands; it’s not useful for modelling system’s thread demands.

Let \( T_a_n \) be the set of \( n \) freelance coincident elements of client program execution:

\[
T_{a_n} = \{t_{a1}, t_{a2}, \ldots, t_{an}\}
\]  

(3.1)

Every transaction \( t_{ai} \) consists of a series of \( e_i \) object requests:

\[
t_{ai} = r_{1i}, r_{2i}, \ldots, r_{ei}
\]  

(3.2)

Let \( x_i (T_{a_n}) \) is single potential interleaving of the client transactions of \( T_{a_n} \). There are several potential interleaving of \( T_{a_n} \), all produces a special global request series \( G ( x_i ( T_{a_n} )) \).

3.3 Problem-Statement
In this segment we use the allusion model of segment3.2 to outline the overall drawback which the apriorism reports.

Specified a collection of \( n \) freelance client thread elements \( T_{a_n} \), we outline the most effective integrated buffer management method collectively which creates the lowest median processing time \( \overline{C}(T_{a_n}) \):

\[
\text{LOW}(\overline{C}(T_{a_n})) = \text{LOW} \left( \sum_{i=0}^{Q_l} c(x_i(T_{a_n})) \right)
\]  

(3.3)

\( Q_l \) is the quantity of potential interleaving’s. \( C((x_i ( T_{a_n} )) \) is the median of the processing time of all of the transactions of \( T_{a_n} \) under the \( x_i ( T_{a_n} ) \) interleaving:
Where \( PT(x_i(Ta_n), ta_j) \) is the processing time of the \( ta_j \) transaction in the \( x_i(Ta_n) \) interleaving. \( n \) is the total quantity of transactions.

The median processing time beneath every potential client-thread interleaving’s is the metric to be optimized. As a result of normally client-threads will begin at any time and lots of system & user aspects can have an effect on the interleaving produced. Thus the buffer-management method established would accomplish fine for a mean of all potential interleaving’s

### 3.4 Integrated-cost-Model

Here we define an integrated-cost-model that includes the results that dynamic-clustering, buffer-replacement, static-clustering, & pre-fetching have on system performance. This cost-model is outlined by exploiting the allusion-model of segment3.2.

The threads sculptured during this segment contain:

- A dynamic-clustering thread (DC)
- This-client thread (TC)
- Opposite-client threads (OC)
- A prefetcher-thread (P)

Equ3.5 illustrates the processing time for the transaction \( t_i= ( r_1, r_2, \ldots, r_e) \) in the \( x_i (Ta_n) \) interleaving.

\[
PT(x_i(Ta_n), ta_i) = \sum_{r=0}^e (IO_{OT}(r) + CPU_{OT}(r)) + \\
\sum_{r=0}^e (IO_{DCR}(r) + CPU_{DC}(r)) + \\
\sum_{r=0}^e (IO_{TCR}(r) + CPU_{TC}(r)) + \\
\sum_{r=0}^e (IO_{PR}(r) + CPU_{p}(r))
\]

(3.5)

The description of the expressions of equation 3.5 is given below:

- The \( IO_{OT}(r) \) &\( CPU_{OT}(r) \) expressions denote to the time this-client was congested because of disk I/O &CPU used by different sources together with opposite-clients & buffer-management:
  - Other disk I/O:
    \[
    IO_{OT}(r) = IO_{OCR}(r) + IO_{BW}(r)
    \]
    (3.6)
IO\textsubscript{OCR}(r): time this-client was congested because of scan I/O completed by opposite-clients among allusions \(r\) & \(r-1\). Dynamic-clustering, static-clustering, buffer-replacement & prefetching have an effect on this expression within the similar manner as IO\textsubscript{TCR}(r).

IO\textsubscript{BW}(r): time this-client was congested because of write I/O started via the BR algo among allusions \(r\) & \(r-1\). Additionally to buffer-replacement, this expression is additionally laid low with suffering from static & dynamic-clustering. The object to page-mapping is described by static & dynamic-clustering which help in determining the polluted pages which are polluted by user applications after they pollute objects. Dynamic-clustering conjointly has an effect on IO\textsubscript{BW}(r) by polluting pages it selects for dynamic reorganization.

Other CPU:

\[
\text{CPU\textsubscript{OC}(r)} = \text{CPU\textsubscript{OC}(r)}
\]  

(3.7)

CPU\textsubscript{OC}(r): time this-client was congested because of CPU movement produced via the opposite-clients among allusions \(r\) &\(r+1\).

- The IO\textsubscript{DCR}(r) & CPU\textsubscript{DC}(r) expressions denote to the time this-client was congested because of disk I/O & CPU handling via the \textit{dynamic-clustering thread}.

IO\textsubscript{DCR}(r): time this-client was congested because of diskette read I/O movement produced via the dynamic-clustering thread among allusions \(r\) & \(r+1\).

CPU\textsubscript{DC}(r): time this-client was congested because of CPU movement produced via the dynamic-clustering thread among allusions \(r\) & \(r+1\).

- The IO\textsubscript{TCR}(r) & CPU\textsubscript{TC}(r) expressions denote to the diskette I/O & CPU resource’s absorbed by this-client to operate the transaction:

IO\textsubscript{TCR}(r): time used by this-client anticipating its specific demanded object to be browse from diskette among allusions \(r\) & \(r+1\). Assume that the demanded object exist on page \(p\), then IO\textsubscript{TCR}(r) matches the percentage of the time that filling page \(p\) has triggered this-client to be congested. It doesn't embody the time used in anticipating the load accomplishment of alternative pages on the server-queue. Within the circumstance that the demanded page is previously within the procedure of getting loaded via one more thread, then IO\textsubscript{TCR}(r) matches the residual load-time. The worth of this expression is tormented via dynamic-clustering, static-clustering, buffer-replacement & pre-fetching. The first goal of those buffer-management methods is to decrease the worth of this expression.
CPU $\tau_C(r)$: time used on the CPU via this-client among allusions $r$ & $r+1$.

- The IO $\pi_P(r)$ & CPU $P(r)$: expression refer to the time this-client was congested because of inappropriate pre-fetch disk I/O & pre-fetch CPU utilization via the pre-fetch thread:
  IO $\pi_P(r)$: time this-client spends congested because of inappropriate prefetch I/O via the prefetcher thread among allusions $r$ & $r+1$. The definition of the prefetch I/O is that a page which is prefetched & not agreeing to the succeeding disk-page demand. The explanation of IO $\pi_P(r)$ doesn't includes advantages of a prefetched page is allusion subsequently the succeeding disk-page load however before it's ejected (segment6.3.3 states this problem). The explanation for victimisation this oversimplified description of IO $\pi_P(r)$ is that this segment is simply meant to administer the scholar an extensive summary of the approach buffer-management technique’s will have an effect on system efficiency. We leave additional detailed study to the advanced chapters.

CPU $P(r)$: time this-client spends congested because of CPU movement instigated via the prefetcher-thread among allusions $r$ & $r+1$.

3.5 Applied Allusion-Models

The allusion model explained in section 3.2 offers terribly correct info to be used within the cost-models. The transaction is too costly to gather in practice. Hence, standard statistical-models centred on stochastic approach are employed in practice. The median program behaviour is captured by these models & they employ it to judge the possibilities of assured occasions resulting in the future. In this apriorism all buffer-management algorithms offered here uses one or added of the allusion-models declared in this segment.

We initially define every allusion-model for the only thread case in this segment. Later, in segment3.5.4 we define exactly how these applied allusion-models may be employed in the concurrent-multithreaded event.

We currently outline certain expressions that may be employed in the rest of the segment. Let $S$ denote the set of N object’s signifying the complete allusion transaction object residents:

$$S= \{1, 2, 3 \ldots N\}$$

(3.8)

Let $R_n$ signify the object level allusion transaction:
The Independent References-model is well accepted as I I D (series of Independent & Identically Distributed random variables), defines the reference transaction arbitrary process. It is an easy reference-model that deliberates reference’s resulting within the reference transaction to be arbitrary-independent actions. At any time t, the probability which an object \( x \) performs in the transaction is stable & alone dependent on \( x \):

\[
\pi(x) = \text{prob}\{R_t = x\}
\]

(3.10)

Where

\[
\sum_{x=1}^{N} \pi(x) = 1
\]

(3.11)

The I I D model can be denoted as an \( N \) row vector of probabilities:

\[
\vec{\pi} = [\pi(1) \pi(2) \ldots \pi(N)]^T
\]

(3.12)

Reference transaction \( R_n \) is used to evaluate the IID model, we use the succeeding impartial evaluator in lieu of every vector element [3] :

\[
\vec{\pi} = \frac{\sum_{t=1}^{n} \delta_{xRt}}{n}
\]

(3.13)

Where:

\[
\delta_{xy} = \begin{cases} 
1 & \text{if } x = y \\
0 & \text{otherwise}
\end{cases}
\]

The facts for the IID model can be gathered effortlessly, & enforce minimum burden. To gather the facts needed to adumbrate \( \pi(x) \), one simply wants to calculate numeral of periods the object \( x \) is referenced. \( N \) is the count of objects in the transaction, where the storage cost of IID is \( O(N) \).

3.5.2 Simple-Markov-Chain-Model

The simple-Markov-chain-model (S M C) defines the allusion transaction as a Markovian method. The dependency drives back only single component in time because a simple-markov-
chain is an arbitrary series. The probabilistic future behaviour of the method is dependent solely on the current state of the method & isn’t changed by the previous history. In the situation of our allusion transaction at some time \( t \) the probability that an object \( y \) seems in the transaction is stable & solely depends on \( y \) & the formerly demanded object \( x \):

\[
P(x,y) = \text{Prob}\{R_t = | R_{t-1} = x \}
\]

(3.14)

Where

\[
\sum_{y=1}^{N} P(x,y) = 1
\]

(3.15)

An \( N \times N \) matrix of conditional probabilities is used to express the SMC model:

\[
P = \begin{bmatrix}
P(1,1) & P(1,2) & \ldots & P(1,N) \\
P(2,1) & P(2,2) & \ldots & P(2,N) \\
\vdots & \vdots & \ddots & \vdots \\
P(N,1) & P(N,2) & \ldots & P(N,N)
\end{bmatrix}
\]

(3.16)

The reference transaction \( R_n \) is used to evaluate the S M C model, we use the succeeding impartial evaluator for each matrix element [3]

\[
\hat{P}(x,y) = \frac{\sum_{t=2}^{N} \delta x R_t - 1 \delta y R_t}{\sum_{t=1}^{N} \delta x R_t}
\]

(3.17)

The tagged directed graph \( G (P) \) can be used to represent the S M C matrix \( P \): the vertice’s denote the object’s; the edges are tagged with elements of \( P \). i.e., an edge from node \( x \) to node \( y \) is tagged via \( \hat{P}(x, y) \). During this circumstances, the allusion transaction is taken into account because the effects of an arbitrary march on graph \( G (P) \). \( G (P) \) states the “normal” behaviour of the program.

The cost of storage of I I D model is very less in comparison to the S M C model, since the frequency is stored by S M C model by which every pair of objects is retrieved one after the other. The storage cost of IID is \( O(N) \), where \( N \) is the quantity of object’s in \( G (P) \). Whereas, the storage cost of S M C is \( O(E) \), where \( E \) is the quantity of edges in \( G (P) \).

### 3.5.3 Higher-Order-Markov-Chains-model

The generalization of higher-order-Markov-Chains-model (HOMC) model is the S M C model to a \( k \)-th order stochastic-process. In comparison to the simple markov-chain model the higher
order markov-chain-model future behaviour is solely dependent on the previous k states. In the situation of our allusion transaction at some time \( t \), the probability that an object \( y \) looks in the transaction is stable & solely dependent on \( y \) & the k lately demanded objects. Though higher-order-markov-chain-model is extra correct, it includes storing data for series of \( k \) objects allusions. At object-grain this model is unfeasible to utilize because of huge statistics storage constraint. But higher-order-markov-chain-model is extra feasible at page-grain. A number of the pre-fetching algorithm’s advised within chapter-6 utilize HOMC model on the page-grain.

3.5.4 Concurrency
Previously the Applied Allusion-models are defined for the individual threaded circumstance. The problems near the use of the applied allusion-models in a concurrent multithreaded situation are detailed in this segment.

3.5.4.1 Independent-Reference-Model
This model takes up reference’s appearing in reference-transaction to be arbitrary independents actions. So IID doesn't model sequential dependence. Therefore any interleaving of N simultaneous thread processing generates the equivalent effective global IID model. This fits the intention of this apriorism splendidly. The explanation is that IID is employed during this apriorism to model global system behavior for creating optimization choices which profit median program behavior (usual performance of every probably interleaving’s). This agrees with the problem statement as given above in this chapter; we have an interest to discover buffer-management techniques that offer the most effective median presentation of all probable thread interleaving’s.

3.5.4.2 Markov-Chain-Models
As we have seen above that IID is unable to model the sequential dependency among allusions but the Markov-chain-models (MCM) is able to model the sequential dependency among allusions. Simply implementing Markov-chain-models to the global reference stream (generate from one specific interleaving) effects in generating a Markov-chain-models model that solely exhibits the features of that specific interleaving. This trusting utilization of Markov-chain-models is not attractive since we utilize Markov-chain-models to model median programming
behavior which should exhibit a mean of all potential interleaving’s (as in the problem statement section above). In distinction, exploiting Markov-chain-models to the transaction produced by all thread distinctly & then combining the alone models to generate 1 global leads to the similar model irrespective of the interleaving. This method picks up the sequential dependencies among references appearing in the similar thread transaction which is independent of the specific interleaving. The algos used by Markov-chain-models in this apriorism are outlined to offer finest median program performance (median of all probably interleaving’s).

3.6 Conclusion
This affiliate outlined the system-models which will be the topic of research during this apriorism. Moreover the allusion-model utilized via the cost-models of this apriorism was too defined. An integrated-cost-model was defined. The overall mechanism of integrated-cost-model helps in defining prospects for refining buffer-management techniques. Methods for optimizing the integrated-cost-model via manipulating the interactions among static-clustering & dynamic-clustering is shown in following affiliate.