SELF TUNING MEMORY MANAGEMENT FOR DATA SERVERS

By Sangeetha Sivaprakasam
Introduction:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

What is memory tuning?

When you run multiple instances on a computer, each instance dynamically acquires and frees memory to adjust for changes in the workload of the instance.
Need for memory tuning:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web-based systems.
8) Conclusion.
9) Bibliography.

- In case of complex software.
- In case of data server in multi-user mode and multiple data-intensive decision support queries.
- Increasing data volumes and critical decision.
- Thrashing, memory bottle, Memory contention neck.
- Automatic tuning decisions reduce the cost of human administration.
Self – tuning server caching:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self – tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- Memory in data server is for caching frequently accessed data to avoid disk I/O.

- Cache manager is to maximize the cache hit ratio.

- The most used replacement is LRU (Least Recently Used) algorithm.

  a) Sequential scan over large set of pages.

  b) Random access to pages sets with highly skewed cardinalities.
Self – tuning server caching:

To overcome these deficiencies – had developed – no of tuning methods but they are not fully self – tuning.

The various approaches are:

1) **PANDORA**:

   - This approach relies on explicit tuning hints from programs.
   - This is an hint processing approach. Eg: a query processor engine.
   - The difficulty is hinting passing approach is very limited and bears high risk.
Self – tuning server caching:

1) Introduction.
2) Need for memory tuning.
3) Self – tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self – tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

**SISYPHUS**:

- This approach aims to tune the cache manager by portioning the overall cache into separate “Pools”.
- It works well with partitioning index Vs data pages.
- But the difficult - appropriate pool size and proper assignment of page classes of pools.

**SPHINX**:

- It abandons LRU and adopts a replacement policy based on access frequencies.
- LFU (Least frequently used ) policy – optimal for static work load ----pages have independent reference probabilities.
1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- The problem in sphinx can also be improved by using a “Nike approach” - LRU-k algorithm.

- It uses three methods observe-predict – react.

Observation:

- It keeps limiting on relevant page’s reference history – k last reference time points.

- ‘Relevant’ - all pages that are currently in the cache plus some more pages that are potential caching candidates.

- Five - minute rule - last 5 mins can be safely discarded.
Self – tuning server caching :

1) Introduction.
2) Need for memory tuning.
3) Self – tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self – tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

Predictions :
• Page’s specific access rate is known as page’s heat.

• Page’s heat(p) = k / now – tk.

• Probability for accessing the page within next T time units is
  1 - e ^ - (heat(p) * T).

• Optimal to rank pages - near-future access probabilities.

Reaction :
• When page - freed up in cache LRU-k algorithm replaces the
  pages with smallest value for above estimated probability.
Self – tuning server caching:

1) Introduction.
2) Need for memory tuning.
3) Self – tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self – tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- This algorithms can be generalized with variable size caching (documents) rather than pages.
- We calculate temperature of document.
- Caching documents are simply ranked by their temperature.
Automatic tuning of server and cache memory:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- A data server needs to manage also working memory for long running operations.

- Memory management should not focus on single global performance.

- It has consider to different workload classes.

- System cannot automatically infer importance of each class - needs human administrator.

- Mechanism for handling multiple work load classes - class specific memory areas.

- The partition is merely conceptual and not physical - memory area - shared by multiple workload classes.
• Approaches for automatic memory performance is described as a feedback loop.

Uses moving time window averaging. Here observation widow must be carefully choosen.

An algorithm is used to predict the performance change and so response time predictions are concerned i.e., is Ri of class i as function of M1,…Mm memory areas. Approx Ri(M1,…Mm) is difficult.

Re-initiate prediction is found by max(Ri / Gi,1<=i<=m) where Ri is response time and Gi is response time goal of class i.
Exploiting distributed memory:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web-based systems.
8) Conclusion.
9) Bibliography.

Two cases:

- High end data servers implemented on server clusters.
- Collection of independent servers with data replicated across all of them.
- Distributed caching algorithm—controls dynamic replication of data objects in (fixed sized pages or dynamic documents) caches.

Two approaches:
1) egoistic caching.
2) altruistic caching.
Exploiting distributed memory:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

Egoistic:

- Each server runs on local cache replacement algorithm – LRU and LRU-k.
- It views remotely cached data that is not locally cached.
- It ends with hottest data fully replicated and in all caches with little space left out for others.

Altruistic:

- It aims at maximizing this replication by giving preference in the local cache replacement to data.
- That data should not be cache resident in different server.
Exploiting distributed memory:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self – tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- For high band width network altruistic approach is better – affordable overhead.
- In fastest interconnect it becomes congested under high load.
- Mathematical cost model - it decides which method is useful under the current workload and system settings.
- Benefit is proportional to mean response time of data and requests over all servers.
- This model includes disk queuing the entire approach can even contribute to disk load balancing.
1) Introduction.

2) Need for memory tuning.

3) Self-tuning server caching.

4) Automatic tuning of server and cache memory.

5) Exploiting distributed memory.

6) Integrating speculative prefetching with caching.

7) Self-tuning caching and prefetching for web based systems.

8) Conclusion.

9) Bibliography.

- Caching reduces overall disks I/O load.

- To reduce response time prefetching is used.

- Prefetching brings relevant data into memory already before it is explicitly required.

- It pays off well - high latencies data request.

- It is beneficial with a certain probability like in case of sequential scans not in case of near access patterns of ongoing operations or client sessions.
Integrating speculative prefetching with caching:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- Alternative method is to access near future access probabilities - stationary heat statistics or corresponding temp value.

- The method is temperature based vertical data migration in.

- It keeps a list of the top temp non cached data units and considers their prefetching in desc order of temperature.

- Prefetching is initiated only when the corresponding documents temp exceeds the temp of the documents.

- When latencies of fetching non-cached documents vary cost benefits consideration should be further refined explicitly.
Integrating speculative prefetching with caching:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- With length T the expected number of access to document d within time ‘T’ is
  \[ N_{\text{spec}}(d) = \text{heat}(d) \times T \]

- Benefit of prefetching document
  \[ d = \frac{N_{\text{spec}}(d)}{\text{size}(d)} \times \text{Fetch}_{\text{time}}(d,v) \]

- Where \( \text{Fetch}_{\text{time}}(d,v) \) is the estimated time for accessing d on its “home location”.

- Where v can be secondary storage, an online volume in tertiary storage or offline volume.
1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- The division by size(d) is normalization per cache space unit.

- This method is for aggressive prefetching and not for speculative.

- Here overhead is low comparable to LRU-k bookkeeping.
Self – tuning caching and prefetching for web based systems:

1) Introduction.
2) Need for memory tuning.
3) Self –tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self – tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

- When servers are accessed over the web or use tertiary storage incur very high latency.

- Stochastic prediction for near future requests must be more “aggressive” but needs to be more “accurate”.

- A richer class of models used is Markov chains.

- Markov chain based algorithm has been investigated for prefetching and caching.

- In prior methods they focussed on reference pattern of a single client and assumed discrete time.
Self – tuning caching and prefetching for web based systems:

1) Introduction.
2) Need for memory tuning.
3) Self –tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self – tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

• McMin (Markov-chain based Migration for near line storage) - different interaction speed of clients - CTMC.

• In web based access to a digital library – CTMC captures variability.

• It is possible to compute both the expected number of near future access to a document d, Nspec(d) - appropriate precomputations.

• The (d,Nspec(d)) both of these values can be aggregated over multiple CTMC models one for each active client session and “arrivals”,”departures” as separate sessions.
Conclusion:

1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web-based systems.
8) Conclusion.
9) Bibliography.

- The methods - geared for centralized, high-speed interconnected and widely distributed data servers.

- The common method we followed is:
  - Observation – online statistics
  - Prediction – mathematical models
  - Reaction – feedback loop

- Space need for online statistics must be carefully controlled.

- CPU time overhead of predictions may be a critical factor.

- Self-tuning algorithms will penetrate products and contribute towards zero-admin and trouble-free servers.
1) Introduction.
2) Need for memory tuning.
3) Self-tuning server caching.
4) Automatic tuning of server and cache memory.
5) Exploiting distributed memory.
6) Integrating speculative prefetching with caching.
7) Self-tuning caching and prefetching for web based systems.
8) Conclusion.
9) Bibliography.

Bibliography:

- Goal oriented buffer management revisited SIGMOD conf., 1996 --- Brown, K., Carey, M., Livny, M.,

- Adaptive database buffer allocation using query feedback VLDB conf., 1993 --- Chen, C.M., Roussopoulos, N.,

ANY QUESTIONS

Thank you