Chapter Two
Chapter 2

SSCMU: Software System for Cache Memory Utilization

1. Design Issues of SSCMU

a) Introduction

As the gap between the Processor speed and memory speed increases, the memory performance has become an important factor for overall performance of processing. It is challenging task to optimize the memory performance for diversity, complexity and runtime dynamics of database systems as well as memory hierarchies.

In EASEDB: A Cache Oblivious Memory Query Processor [1], Cache oblivious query processor for in-memory relational query processing that shows optimization of cache performance and overall performance of query processing. It developed visualization ratio interface to show detailed performance of EASEDB. It is implemented in JAVA SWT. The importance of EASEDB is that it shows comparison with cache
memory counterpart with both parameter value in cache algorithms and hardware platforms varied.

In A Novel Asynchronous Software Cache Implementation for the Cell-BE Processor [3] describe the implementation of a runtime library for asynchronous communication in the Cell BE processor. The run time library implementation provides with several services that allow the compiler to generate code. The library implementation is organized as a software cache and replacement; data write back, memory synchronization and address translation. The important feature of this software is, it guarantees that all those services can be totally opportunities to the compiler to organize the generated code in order to overlap as much as possible computation with communication.

Software Cache on the Cell BE Processor [2], it gives programmability and performance in certain applications such as those with irregular memory references on multi-core architectures like the cell processor where on chip memory is a precious resource. It presents the performance of a histogram application using the software cache. It propose a static analysis tool that takes memory references generated by a given application and gives the optimal cache parameters that can be used to configure the software cache for the application to run on the Cell Broadband Engine. The important feature of this software is that it also presents the results of analysis done with this tool for the trace generated by a heap sort application

We propose to demonstrate Software System for Cache Memory Utilization: SSCMU, cache-oblivious algorithms for in-memory relational processing. The cache
oblivious notion from the theory community refers to the property that no parameters in
an algorithm or a data structure need to be tuned for a specific memory hierarchy for
optimality. As a result, SSCMU automatically optimizes the cache performance as well
as the overall performance of query processing on any memory hierarchy. We have
developed a visualization interface to show the detailed performance of SSCMU in
comparison with its cache algorithm, with both the parameter values in the cache
oblivious algorithms and the hardware platforms varied.

b) Feasibility

SSCMU is new cache oblivious memory utilization that optimize cache oblivious
algorithm on any multilevel memory hierarchy. In SSCMU, the data structure and
algorithm for processing do not depend on any cache parameter of specific memory
hierarchy, for example, the cache capacity and block size. They automatically achieve a
high performance comparable to the fine tuned cache algorithms on various platforms.

c) Study of Prototype

SSCMU optimizes the performance of processing on levels of memory hierarchy.
This is especially desirable for the levels of above the main memory, because cache at
these levels, for example, L1 and L2 cache are managed by hardware. As a result, as the accurate state in information of these caches is difficult to obtain due to system runtime dynamics and the hardware complexity, we consider cache oblivious algorithms, that can automatically improve the memory performance of processing.

The system demonstration express cache behaviors of cache oblivious processing on memory hierarchy in comparison with cache processing and also provides an intuitive way of visualizing the cache performance as well as overall performance of processor. Through our visualization tool, we can examine with status of execution at runtime through demonstration. We show that

1) Cache obvious algorithms optimize the level of memory hierarchies.
2) Cache oblivious algorithm in SSCMU and its overall performance is fine tuned to cache behavior.

2. Stages of System Design

To solve actual problems, a developer must incorporate a development strategy that encompasses the process, the methods and tools, layers. Because software is always part of larger system, work begins by establishing requirements elements and then allocating some subset of these requirements to software. This system is essential when software must interact with other element such as hardware, people and databases.
a) Requirement Analysis

SSCMU is intensified and focused specifically on software. To understand the nature of programs, for software, as well as required function, behavior, performance and interface. Requirements for both the system and software are documented and reviewed with algorithms.

Software requirement analysis may divide into five areas 1) Problem recognition 2) Evaluation 3) Modeling 4) Specification 5) Review. It is important to understand software in a system and review the software scope that was used to generate cache oblivious algorithms.

Problem recognition and Evaluation is the major area of effort for analysis. It must define all observable data, evaluate the flow and content of information, define function, and understand cache behavior in context of event that affect the system, establish cache oblivious interface characteristics, additional design constraints. Each of these tasks serves to describe the problem so that an overall approach or solution may synthesize
b) Architecture

SSCMU is multistep process that focuses on four distinct attributes of program; data structure, software architecture, interface representation and procedural detail. The design process translates requirements into representation of software that can be accessed before caching begins. Like requirements, the design is documented and becomes parts of software configuration as shown in fig1.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{system_architecture.png}
\caption{The System Architecture of SSCMU}
\end{figure}
There are three components in MATLAB namely MEX function, Execution Engine and GUI. In MEX function, c file consist of cache oblivious algorithms. The execution engine is divided into three layers coding, analysis and testing. All these components and algorithms are designed to be cache oblivious.

We use two main methodologies, divide and conquer and recursive process, to develop the cache-oblivious algorithms such as cache oblivious matrix multiplication, cache oblivious transposition, dynamic programming, sorting and B-tree.

c) Visualizing the cache performance

The visualization tool is a graphical user interface (GUI) that dynamically displays the performance of cache oblivious and the state of all cache behavior in the system. The GUI is implemented in MATLAB7, since MATLAB is widely applicable on various systems. The performance on CPU caches including the numbers of cache hits and misses are typically measured using a profiling tool.

Fig 2 shows the main execution window when simple flow is running. There is main window of software and on the right of window, there is sequence flow of cache oblivious matrix multiplication; the result is automatically obtained from MATLAB environment.
The user can choose cache oblivious algorithms to execute the methods or techniques. When the cache oblivious algorithm is chosen, its parameters can be individually configured. More specifically, when MEX function can be selected in form of a test button. The default parameter values for the cache oblivious algorithm, cache capacity and cache line size. According to the algorithmic characteristics, they can be fit. Since cache is typically a major bottleneck for memory accesses. The user also click on analysis button and view the application and execution time slots.
d) Why MATLAB?

1) High-level language for technical computing

2) Development environment for managing code, files, and data

3) Interactive tools for iterative exploration, design, and problem solving

4) Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration

5) 2-D and 3-D graphics functions for visualizing data

6) Tools for building custom graphical user interfaces

7) Tools for building custom graphical user interfaces

8) Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java™, COM, and Microsoft® Excel

The key feature of MATLAB is

---

*S.R.T.M. University Nanded*
1) Developing Algorithms and Applications

MATLAB provides a high-level language and development tools that let you quickly develop and analyze your algorithms and applications. The MATLAB language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables fast development and execution. With the MATLAB language, you can program and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks, such as declaring variables, specifying data types, and allocating memory. In many cases, MATLAB eliminates the need for ‘for’ loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code. At the same time, MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, object-oriented programming (OOP), and debugging features.

2) Development Tools

MATLAB includes development tools that help you implement your algorithm efficiently. These include the following
a. MATLAB Editor

MATLAB is not only a powerful language, but also features an excellent Integrated Development Environment, including a powerful editor, debugger, profiler, and compiler. It is the recommended way to interact with MATLAB. MATLAB commands are executed either at the command prompt or by running scripts or functions, which can be created and edited with the built in editor. To launch the editor, if it is not already open, type edit or edit filename. Commands can be entered here and executed as a script. They are saved with a .m extension. To run your script, type in the name at the command prompt, or press F5 or the save and run button at the top of the editor. Functions can be written here as well, as discussed here. User can set break points to halt execution at certain lines for debugging.

b. Code Analyzer

A source code editor is a text editor program designed specifically for editing source code of programmers. It may be a standalone application or it may be built into an integrated development environment (IDE). Editors have features specifically designed to simplify and speed up input of source code, such as syntax highlighting, auto complete and bracket matching functionality. These editors also provide a convenient way to run a compiler, interpreter, debugger, or other program relevant for software development process. So, while many text editors can be used to edit source code, if they do not
enhance, automate or ease the editing of code, they are not source code editors, but simply text editors that can also be used to edit source code.

c. MATLAB Profiler

MATLAB provides a nice performance and tuning tool called profile. This allows users the ability to observe actual computation times for a given program. There are two ways to use the profiler, either function by function or running the whole program. Older versions of MATLAB only allow for profiling specific functions, this means you will need to individually profile each called function to get an overall time. If a newer version is used (Newton lab), MATLAB will output timing for all the functions used during a run, this includes all of the called functions. The output can then be viewed in html form. There is a great resource on the web, containing explanation and examples.

d. Directory Reports

MATLAB Report Generator can take any information from your MATLAB workspace and export it to a document in the form of a report. In this phase, we look at how to work with MATLAB Report Generator. We will open an existing report setup file and add and remove components from it. We will look at several of the components, and then create a
report in HTML format. We will open the report file and walk through it. We will create a new style sheet by modifying an existing style sheet. Then, we will create a PDF report, using the new style sheet.

e. Designing Graphical User Interfaces

A graphical user interface (GUI) is a graphical display in one or more windows containing controls, called components, which enable a user to perform interactive tasks. The user of the GUI does not have to create a script or type commands at the command line to accomplish the tasks. Unlike coding programs to accomplish tasks, the user of a GUI need not understand the details of how the tasks are performed. GUI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders—just to name a few. GUIs created using MATLAB Tools can also perform any type of computation, read and write data files, communicate with other GUIs, and display data as tables or as plots.

e) DEMONSTRATION
We use MATLAB visual tool to study cache oblivious algorithm in SSCMU in comparison with cache algorithms. We will run different cache oblivious algorithm and their different parameters are listed in Table 1.

**Table 1: Cache Oblivious Algorithms**

<table>
<thead>
<tr>
<th>Sr no</th>
<th>Algorithms</th>
<th>Parameters</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cache Oblivous Matrix Multiplication</td>
<td>3 x 3</td>
<td>Sequential Access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 x 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 x 5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Cache Oblivous Matrix Transposition</td>
<td>3 x 3</td>
<td>Sequential Access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 x 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 x 5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Dynamic Programming</td>
<td>3 x 3</td>
<td>Simple</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 x 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 x 5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Cache Oblivous Matrix Sorting</td>
<td>Bubble Sort</td>
<td>Sequential Access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KK sorting</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Cache Oblivous Matrix B-Tree</td>
<td>DFS</td>
<td>Sequential Access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BFS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>In-order</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Van-Emde Boas</td>
<td></td>
</tr>
</tbody>
</table>

We will vary different cache oblivious algorithm with their different parameters using SSCMU as shown in fig 3. It gives sequence flow SSCMU Software.
Fig 3 shows Execution flow of cache oblivious algorithm in SSCMU Software. There are five major components, namely Multiplication, Transposition, Dynamic Programming, Sorting, B-tree. Each component consists of different methods. User may test different parameters using given methods. All these components and algorithms are designed for cache oblivious algorithms.
3. Modules

SSCMU executes Cache Oblivious Algorithm that designed to perform well, without modification on multiple machines with different cache sizes or for a memory hierarchy with different levels of cache having different sizes. The idea of cache oblivious algorithms was conceived by Charles E. Leserson[4] as early as 1996 and first published by Harald Prokop[4] in his master thesis at the Massachusetts Institute of Technology in 1999. Optimal Cache Oblivious Algorithms are known for Cooley-Tukey FFT algorithms, matrix multiplication, sorting, matrix transposition and several other problems. The goal of SSCMU is to reduce the amount of tuning that is required. Cache Oblivious Algorithms are analyzed using idealized model of cache known as cache oblivious model.

In SSCMU, we designed following cache oblivious algorithms

1) Cache Oblivious Matrix Multiplication
2) Cache Oblivious Matrix Transposition
3) Dynamic Programming for Cache
4) Cache Oblivious Sorting
5) Cache Oblivious B-Tree
### a. Cache Oblivious Matrix Multiplication

Matrix multiplication is one of the most common operations on matrices, often used as kernel operation for other algorithms. In SSCMU we designed different cache oblivious matrix multiplication methods such as fig 4.

![Fig 4: SSCMU: Cache Oblivious Matrix Multiplication Algorithms](image)

Fig 4 shows SSCMU execution flow of cache oblivious matrix multiplication and their new method with their sequence flow. We design new cache oblivious matrix multination techniques as

#### i. Multiplication Using Sequential Access

In this method, we will present an approach that uses a sequential access processing. The sequential access technique that converts the two dimensions into single. It is very easy to manipulate different matrix transformation operations.
ii. **Matrix Multiplication Using Recursive Storage**

We will present an approach that uses a Recursive Storage access processing. It totally avoids jumps in access to all matrices involved and reduces cache miss. It also uses only two types of recursive blocks.

iii. **Matrix Multiplication Using Locality Preserving**

We will present an approach that uses an ordering of matrix element that is based on Peano space filling curve [M. Bender et. al. Elsevier Sci. Dec 2004]. It gives results form a recursive constructing idea. So our approach is similar to many recursive multiplication schemes. However our presented scheme follows column-major execution process and shows optimal spatial locality in that sense. After each individual column-major multiplication operation the next elements to be access will direct next column elements of previous ones.

iv. **Matrix Multiplication Using New Approach**

In this method, we will present an approach that uses a Row-major recursive Storage access processing. However, our presented scheme executes natural process of matrix multiplication in access to all matrices involved and reduces cache miss. It is very easy to manipulate different matrix transformation operations.
v. Matrix Multiplication Using Z curve

We will present an approach that uses a Z Curve Oblivious Recursive Storage access processing. The Recursive Storage access technique that automatically achieve fine-tuned algorithms on a multi-level memory hierarchy, we simply change the execution flow of sequence with different blocks.

In SSCMU Cache Oblivious Matrix Multiplication Module algorithm design as list of Sequence Access, Locality Preserving, New approach, Z-curve and Recursive Storage. It sequence format as shown in screen shot fig 5,

![Menu of Cache oblivious matrix multiplication](image)

Fig 5: Menu of Cache oblivious matrix multiplication

In this phase, User may choose any one method from given list; it then executes the in format of three sub modules as Sequence flow, Coding and Testing. Each Cache Oblivious Algorithm will be process in this sequence format. Sub modules of every cache oblivious algorithm screen shot may be in fig 6.

*S.R.T.M. University Nanded*
b. Cache Oblivious Matrix Transposition

Matrix transpositions are a fundamental operation in linear algebra and in Fast Fourier transforms and applications in numerical analysis, image processing and graphics. In SSCMU we have designed cache oblivious matrix transposition using sequential processing as shown in fig 7.
Fig 7: SSCMU: Cache oblivious matrix transposition processing

Fig 7 shows sequential flow of Cache oblivious matrix transposition algorithm. In this phase, user may test cache oblivious sequential access matrix transposition method. It will applicable for different parameter 3 by 3, 4 by 4 and 5 by 5. The method may divide sub module as Matrix transposition coding, analysis and testing as shown screen shot fig 8.

Fig 8: Sub Menu of Cache oblivious matrix transposition
c. Dynamic Programming for Cache

A dynamic programming is a solution for special type of application such as context free language recognition matrix chain multiplication and optimal binary trees. In SSCMU we have test three dynamic algorithms as Diagonal Algorithms, Horizontal Algorithm, and Vertical Algorithm a shown in fig 9.

In this phase, User have to choose Dynamic Programming algorithm from given list, Each algorithm may processed 3 by 3, 4 by 4, 5 by 5 matrix parameters. It may divide as submenu analysis and testing with concern sequence. The screenshot may appear as submenu in fig 10.
d. Cache Oblivious Sorting

A simple introductory example of the application of the incompressibility method is sorting. In SSCMU we designed Cache oblivious sorting using sequential access. The Main menu of SSCMU shows in fig 11.
In this phase, User may process Cache oblivious Sequential sorting. It gives algorithmic approach for new method. It may tested and compare with famous Bubble sort method. It may divide as submenu analysis and testing with concern sequence. The screenshot may appear as submenu in fig 12.

![Image of Cache Oblivious Sorting Menu](image)

*Fig 12: Sub Menu of Cache Oblivious Sorting*
e. Cache Oblivious B-Tree

Cache Oblivious B-tree are designed to achieve good data locality at one level of the memory hierarchy and for one fixed block size. In SSCMU we designed different cache oblivious B-tree using sequential access. The menu of SSCMU screen shot fig 13.

In this phase, User may process Cache oblivious B-tree Sequential Processing. It gives algorithmic approach for new method. It may tested and compare with famous DFS, BFS, In-order and Van Emde Boal layout structure. It may divide as submenu analysis and testing with concern sequence. The screenshot may appear as submenu in fig 14.
In SSCMU each cache oblivious algorithm is executed in Design, coding and testing sequence sub menu method. It also describes the timeslots and design format of different methods of cache oblivious algorithms.
4. REFERENCES

[21] Bingsheng He Yinan Li “EASE DB : A Cache Oblivious in Memory Query Processor “(Hong Kong Univ. of Science and Technology) {Seven luo} @ cse.ust.hk ACM SIGMOD 07 June 12-14 2007 Beijing China.


[23] Jairo Balart, Marc Gonzalex, XavierMartorell ‘Novel Asynchronous Software Cache implementation for the cell BE processor’ Barcelona Supercomputing Center(BSC) Technical University of CATAunya(UPC) IBM Watson Research centre 2005