GAMESDB:
A Virtual Memory System for GAMESMAN

Evan Huang

GamesCrafters Research Group

University of California, Berkeley

TABLE OF CONTENTS

1. Update History

12/30/2006 – Initial Version.

2. Prerequisite Knowledge

Terminology used in this document should be standard in describing virtual memory mechanisms in computer science. Many university introductory computer science courses, as well as enormous literature on operating systems, explain the meaning and philosophy behind such devices to mimic (almost) large amounts of memory beyond the limit of what is physically available.

A typical reader of this document should be familiar with C/C++, and understands the meaning of relevant terms and concepts of virtual memory. A fellow GamesCrafter, who is a UCB undergraduate and taking/having taken the class of Computer Science 61C, satisfies that requirement.

3. Overview

This document is intended to introduce the particular implementation of a virtual memory subsystem in GAMESMAN, the game tree traversal and analysis engine made by the GamesCrafters Research Group in the University of California, Berkeley. It is especially intended for a fellow GamesCrafter programmer who wants to understand the architecture of the subsystem in order to improve it or apply that knowledge to anything/anywhere he pleases.

The subsystem started as an effort to provide disk-based storage functionality to GAMESMAN, because of the sizes of the game trees are too big to be stored entirely inside physical RAM. In the course of its implementation we realized that many of the techniques, including paging and caching for the content on disk to speed up access, were the same as those utilized for virtual memory management in modern operating systems.

This document serves as a high-level description of the subsystem. Copious comments sprinkled throughout the actual code tell the purpose of various elements and should be reviewed along with the code to get a better picture of the subsystem's inner workings.

4. Usage

All files that implement the subsystem, in addition to a Makefile template (Makefile.in) and a sample testing harness (dbtest.c) are available in the directory gamesman/src/core/filedb. They take advantage of GAMESMAN's build infrastructure and integrates itself into GAMESMAN as a selectable database.

To use the filedb/gamesdb functionality, use the --filedb switch, combined with other switches to disable conflicting functions (--nobpdb comes to mind, but this should change in the near future).

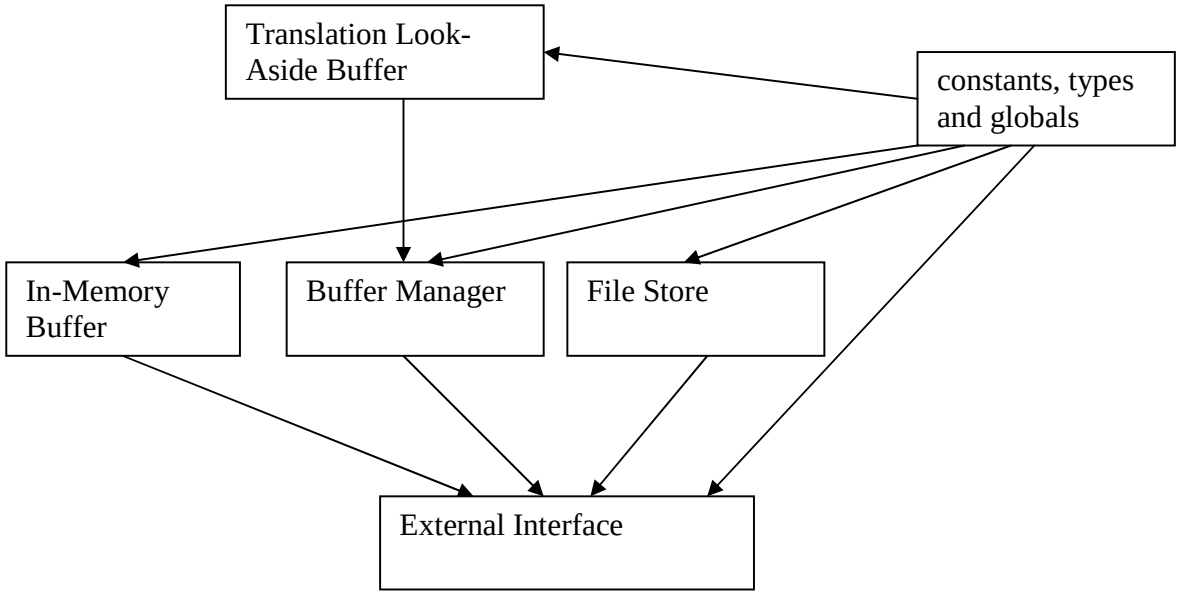
gamesdb should really be further pushed to the backend so that it only interacts directly to routines that handle the database interface with the rest of the program.

5. Implementation

5.1. Architecture Overview

5.1.1. System Call Graph

This is a rough sketch of what the major components are and which calls which:



5.1.2. Constants, Types and Globals

This includes db_types.h and db_globals.h. They define the data structures and constants used in all other components of the subsystem.

5.1.3. The In-Memory Buffer

This includes db_buf.c and db_buf.h. They allocate, read, write, and free the memory space created to cache frequently accessed disk pages in the hope to speed up access.

5.1.4. The Translation Look-Aside Buffer (TLB) for the In-Memory Buffer

This includes db_basichash.c and db_basichash.h. It is a simple implementation of a directly mapped hash table that has a fixed number of buckets with chains for collisions. It

serves as an inverted page table to keep track of the mapping of physical pages in memory and those on disk. We realize the potential inefficiency with long chains and is made the routines to move up recently accessed elements to the front of the chain, although it is hard to estimate the gains from that modification.

5.1.5. The Buffer Manager

The heart of page table management, this block of code maintains book-keeping information, including dirty and valid bits, as well as the TLB.

5.1.6. The File Store

This code takes care of swapping pages in and out of the physical memory and onto the disk. They are in db_store.c and db_store.h.

5.1.7. The External Interface

This is the glue code that takes a get/set call from the outside and orchestrates the operations of various components inside the subsystem. They are housed inside db.c and db.h. Interestingly they are the same as those used in a normal GAMESMAN database, and use the same API as the latter. This shows the roots of the abstraction layer that came from the normal database design.

5.2. Naming Conventions

All types, variables and functions bear names prefixed by “gamesdb” to ensure no clashes with the rest of the GAMESMAN core. In fact, gamesdb compiles completely itself without external dependencies.

The pages are saved on disk in the gamesman/bin/data directory. A directory will be created using the name given to the initialization routine, and pages are saved in files N.dat, where N is the id (gotten using the memory address/the number of records inside a page).