Book Proposal
Understanding The Linux Virtual Memory Manager

Mel Gorman

July 9, 2007

Contents

1 Original Proposal

This is a slightly updated copy of the original proposal that was submitted. The vast majority of the information here remains the same and is included for reference.

1.1 The Proposed Book

Proposed Title: Understanding the Linux Virtual Memory Manager

Author: Mel Gorman, currently a Junior lecturer in Operating Systems with University of Limerick, Ireland

Summary of Book: The book gives a detailed description of the Linux Virtual Memory (VM) manager, beginning with an introduction on how to approach reading the code of an open source project. It then describes the theoretical foundation of the VM before giving a detailed description of its implementation in Linux. An appendix provides a source code line by line commentary for a significant percentage of the VM.

Topic Summary: The field of memory management is considered a complex field but descriptions of practical implementations only exist as general descriptions in operating systems books which are insufficient for a thorough understanding. Normally, this would require a person to first study the field of memory management before reading through the implementation line by line which is a considerable task. This book dedicates itself to explaining, in detail, how the memory manager is implemented in Linux cutting down the time needed to understand it from several months to a few weeks.

Unique Approach of Book: First, the book in unique in that it deals exclusively with the virtual memory manager. Other operating system related books try to cover all aspects of the kernel without giving specific focus to one subsystem. The specific books that do exist are usually related to networking.

Second, the introduction chapters are unique in that they do not cover general kernel material, which is adequately covered by other books. They instead describe how to get started in understanding and managing the kernel code with a description of some helpful tools which are included in the companion CD.

Third, the book discusses the theoretical foundations which has been omitted from recent Linux kernel related material but which is of interest to both researchers and developers. However, it also discusses the actual implementation in heavy detail. To make this approachable to readers of various abilities, the books is is split into two "stages". The first stage gives a detailed verbal description with the aid of diagrams and call graphs the architecture of the manager which is suitable for people who need a clear understanding of how it functions. The second stage is a detailed line by line coverage of the VM for readers who need a precise account of how the VM works which is of particular importance when comprehending later implementations but would be far too detailed for the first reading.

Included with the CD is two novel tools. The first is called **VM Regress** which was developed as a framework for analysing, testing and benchmarking the VM. The tool currently is able to perform a number of operations and provides a solid framework for a dedicated user to build new tests, the project is being maintained on my website. The second tool, which I also developed, is a call graph generator for C and C++ which is invaluable in illustrating how code is structured. All the call graphs in the book were generated using the tool and it is actively maintained on my website.

1.2 Marketing Information

Size of Market: The principal target market is developers, researchers and academics who need to understand how the Linux VM operates in detail. For these people, a general kernel book is too "light" just as a device driver developer would not use a general book. An overview misses too many of the subtleties and sometimes omits entire aspects. This is unfortunate in the case of the VM as it is critical to overall system performance. One would think that there would be countless books on the subject but instead, there is none that focus specifically on the Linux VM. Specific documentation is rare bordering on the non-existent and the VM is very poorly understood except by a few core developers. An example of this type of complaint may be seen in the last paragraph of this e-mail; *http://lwn.net/2001/0927/a/am-vm.php3.*

A typical argument about documentation may be seen at *http://kt.zork.net/kernel-traffic/kt20011224_147.html#1* where developers argue about the need for documentation. The core problem is that code is simply too dense to be understood without a conceptual framework to start with. This book seeks to provide that framework which allows a reader to fully understand the 2.4 VM which acts as a very solid foundation to understanding later implementations.

Since the first draft was put online in January, my (incomplete) web logs indicate that there has been 32,000 downloads of the main PDF and 17,000 of the code commentary. The HTML versions are difficult to calculate but 14,450 unique hosts have accessed the main document and 5000 for the code commentary. I believe there is a definite group of people who do not believe that code is documentation and would gladly purchase a book on the subject instead.

Key Topic Coverage: The request for VM documentation is something that continuously arises on Linux kernel related mailing lists. There has been previous attempts by people to write documentation but the efforts generally petered out long before they were completed. A web search with VM, Documentation, Lack, Required etc will show a number of mailing lists postings of people asking for documentation and being referred to other OS books, or a C programming book.

What Problem Does This Book Solve?: It solves the problem of how to approach comprehension of the Linux VM. It begins with approaches to understanding and analysing code and describes the tools I used myself to perform the same task giving them a certain amount of credability as being useful. The CD itself includes the full code commentary as well as copies of the tools I developed during the course of writing the book and a web browsable version of the source code for the reader who is required to understand the VM in intimate detail. I do not believe there is any other book which goes into this level of detail with the reader.

Any other points that reflect why this book will sell well?: It is a comprehensive description of a practical virtual memory manager implementation which I believe fills a hole in the OS book market and particularly with the Linux market.

Profile of Audience: OS researchers, both memory management researchers and lecturers compiling notes for an OS module may use the book to give a description of the practical implementation of a memory management system. This will still be of interest to them even if Linux is not their prime research focus as research sometimes consists of comparisons between existing systems. Kernel developers, both new and experienced, will need it before making any attempt to change how the VM works as simple changes can have unexpected and far reaching side-effects. Kernel developers working on other, seemingly unrelated subsystems, will benefit from reading the book as virtually every subsystem from device to process management is wired to the VM to some extent. The VM is unique in this respect as most of the other kernel subsystems are easily isolated. Finally, kernel development is perceived by many to be the "hardcore" of open source development and many will read it when trying their hand at kernel development.

Conferences: Any of the open source related conferences will have some people interested in kernel development attending. There is also some Kernel specific conferences such as the Ottawa Linux Symposium which would have a number of people interested in the book.

Prerequisites: The reader will need to have some understanding of C and software development. They will need to have read at least one general OS book but it does not have to be Linux specific. Any general UNIX book will cover the knowledge they need to get started. The principal books and papers read for the writing of the book are included in the bibliography for further reading.

Reader Benefits:

- Includes an introduction into the practical side of code comprehension
- Full and detailed description of the principals of the VM foundations
- Detailed line by line coverage of the code

Competition: The principal competition is “Understanding the Linux Kernel” by Daniel P. Bovet and Marco Cesati, published by O’Reilly. This is the general starting point recommended to people who enquire about getting involved with kernel development. However, this does not give specific focus to the VM, missed many important subtleties of the implementation, has little discussion of the theoretical foundation, omits parts of the virtual memory manager totally and has only relatively light discussion of some of the code.

1.3 Sales Information

Retail: Linux Programming/Operating Systems

Corporate: A corporation would benefit if they needed a customised kernel for a particular application.

Academic: Any operating systems course which included memory management would benefit from this as it allows a full VM to be taught and understood within one semester. It would be prime benefit to a lecturer who wished to include practical examples of how memory management theory is implemented in practice.

Direct Mail: The first platform to push it is via *http://www.kernelnewbies.org* which is the starting point of any kernel developer and maintains a recommended reading list. One of the maintainers is Rik van Riel, a long time VM developer who has often complained about the lack of solid VM documentation. The second platform would be Linux Weekly News which has long established itself as the best news point to follow kernel development through the years and publishes announcements of new books of interest to kernel development.

1.4 Production Information

Time sensitive: I am currently in the process of putting together a proposal for a PhD and am authoring two papers that I wish to submit for a conference in October. It will be about 6 weeks before I will be able to give full attention to the book but will be able to start preliminary work before then.

Present State of Project: At the moment, it is essentially complete and all the technical details is there. I am confident that the bulk of the work required for the book is complete and it has been proof read enough for me to me sure it is free of technical errors. The principal work remaining is to write on what is "up and coming" with the 2.6 VM.

Estimated Completion Date: I would estimate it would take about 2 months to complete the up and coming sections as they will be mainly additions to the chapters already there.

Number of Permissions Needed from Other Copyrighted Works: None, I am the sole author.

Estimated Final Book Pages: Right now, the book, including the code commentary, stands at 484 pages on an A4 sheet but is pretty compact on the sheet.

Submission Format: At the moment, it is formatted in LaTeX which easily converts into PDF, HTML or plain text formats, depending on preference. It could be easily imported into MS Word as plain text.

Camera-Ready: LaTeX currently formats the document and macros are heavily used to identify how things like fonts should be rendered. All diagrams are currently in postscript.

Number of	
Line Drawings	13
Graphs	57
Tables	28
Black & White Photographs	0
Color Photographs	0
Total	98

2 Whats New Section

The release date for 2.6 has not been set at the time of writing but it is expected that it will be released in the coming months. To get the most from this book, 2.6 will be introduced so that the reader has a starting point if they need to examine the 2.6 code. Once the addition features and differences have been explained, 2.6 becomes a lot more approachable.

2.1 Format of the Sections

Rather than having a 2.6 chapter, I propose to have a "Whats New" section at the end of each chapter in the book. In this section, the new features will be introduced and the important differences between 2.4 and 2.6 explained. These sections will be between a few paragraphs and a few pages long, depending on the section.

This format allows the 2.6 material to be concisely introduced to the reader without inter-mixing 2.4 and 2.6 together throughout the book or, the other extreme, putting 2.6 all in one large chapter.

I would prefer not to put 2.6 in one large chapter unless I have to. The book goes to some effort in breaking up the VM into small manageable chunks. It would "break" the style if 2.6 was handled as one large lump of text with little regard to context.

2.2 Level of Detail

The sections will be quite detailed. The basic principals of each feature will be introduced as well as a brief description of how they are implemented. As they need to be terse, the sections will presume the reader is already moderately familiar with the 2.4 VM and they will be advised in the preamble to read through the whole book before reading the Whats New sections in detail. A sample Whats New section will be provided on request.

3 Companion CD Enhancements

A companion CD has already been put together for the book although it needs to be updated. The aim of the CD was to provide the book in soft format for easy text based searching and to include the software developed as part of the research for the book. The list of what it currently includes is;

- Web server that runs directly from the CD
- Main book in HTML, PDF and plain text formats
- Code commentary in HTML, PDF and plain text formats
- Copies of **VM Regress**, **CodeViz** and **Patchset**
- Web browsable version of Linux 2.4.20 with LXR

A number of new features may be added to improve the appeal. Some ideas include;

- Searchable function index for code commentary
- Ability to generate call graphs from CD
- Link call graphs to LXR so that call graphs can be "clicked" on
- Copies of all software that **CodeViz** depends on such as **GraphViz** and **gcc**
- Replace UL Logo with Prentice Hall or other more suitable logo

4 Detailed Schedule

This section outlines the basic schedule including each of the deliverables, the estimated time to completion and a brief description of what will be provided at the end of the milestone. I estimate that I will be ready to begin writing the book starting at June 23th. I am giving myself one week for each chapter. Some will take longer than that and others will be completed well in advance but it is difficult to be precise.

Milestone: Preamble and conclusion	**Duration:** 3 days
During this time, I will write the preamble, remove all references to thesis from the book and basically give it a more “book” like layout than it currently has. The conclusion as it stands is a little weak for a book and needs a bit of shaking up.	
Completion Date:	June 25th

Milestone: Code Commentary as Appendix	**Duration:** 7 days
The code commentary currently exists as a separate document to the main book. During this stage, I'll integrate the code commentary into the main book as a series of appendices, update the references to match and write a basic introduction to it.	
Completion Date:	July 5th

Milestone: Introduction	**Duration:** 5 days
The introduction as it currently stands is more suitable as preamble than an actual introduction. It needs to be rewritten to give a basic introduction to some basic Linux memory management concepts such as how the TLB is used and the L1 CPU cache.	
Completion Date:	July 12th

Milestone: Code Management	**Duration:** 5 days
The tools that the code management chapter talks about have been updated and now behave differently than the book indicates. This task should not take too long but I want to talk more about the structure of open source projects in general which will take a few days.	
Completion Date:	July 19th

Milestone: Whats New for Chapters 3-8, 10	**Duration:** 5 days
As it stands, I've written the most of the Whats New for Chapters 3-8 and Chapter 10. I will need time to verify though that the information is still valid on the writing date and update if necessary. It will also take time if the format of the Whats New sections changes from what I currently have.	
Completion Date:	July 26

Milestone: Whats New: Slab Allocator	**Duration:** 5 days
A cursory glance shows the implementation of the slab allocator has changed quite a bit for 2.6. It will take a few days to determine exactly how much of a difference there is and write about it. I believe the basic principals are the same though and it is mainly optimisation work. If that is the case, it will take a lot less time.	
Completion Date:	August 2nd

Milestone: Whats New: Page Frame Reclamation	**Duration:** 5 days
Page frame reclamation has changed quite a lot since 2.4 in terms of implementation but from a cursory glance, it looks like the basic idea behind it remains the same. It should not take more than 5 days to document all the changes that have been made and what they mean.	
Completion Date:	August 9th

Milestone: Whats New: Swap Management and OOM	**Duration:** 5 days
Swap management does not look as if it has changed much but as with the above, I need to confirm that. I don't think out of memory management has changed at all which is why I'm including it here	
Completion Date:	August 16th

Milestone: Companion CD	**Duration:** 5 days
The companion CD as it stands is fairly decent and provides a soft copy of the book and code commentary. I would like to implement a few new features such as a searchable function index and be able to generate call graphs from the CD itself if at all possible	
Completion Date:	August 23rd

Milestone: New chapter: Shared Memory	**Duration:** 8 days
One section that is currently omitted in the book is shared memory as I treated it as an IPC mechanism before with little impact on the VM itself. However, I suspect there is many that would disagree and expect this chapter to exist. I am alloting 8 days to write it	
Completion Date:	September 2nd

Milestone: Index and cleanup	**Duration:** 3 days
The index will need to be double checked to make sure the references are in a logical order. This will be fairly time consuming and may require the index to be broken out into two parts, code related index references and "normal" references. The final cleanup will be a double check to make sure the book is consistent and correct.	
Completion Date:	September 6th