IBM Invention Disclosure Form - NO DEFENSE CLASSIFICATION ITEMS

NEXT

ANAQUA®

Summary

Expand All Collapse All

Record ID	95059214	Last Modified	16 Jul 2018 11:44:57
Invention Reference	P201805440	Invention Short Title	Method to Reduce Huge Page Internal Fragmentation
View	Review Status and Patent Application Details	Invention Status	Awaiting Pre-Ranking

Step 1 - Disclosure

All questions marked with an * are required and must be answered in order to submit this invention disclosure form (IDF) to IP Law for processing.

- Use "Wizard" to move through Steps 1 -4 by clicking the Next button. Diagrams, drawings or files should be attached in Step 2. Co-inventors may edit before you SUBMIT the form in Step 4. After you submit the form, the only thing you can do is add a document to the Documents section for "post disclosure" processing.
- Click the "SAVE" button before leaving Step 1 to ensure you retain the information you have entered. Include a "TITLE" in your IDF as it will help you find your invention on your HOME screen when saved.
- All questions are required (i.e., 'N/A' should be selected if applicable). Invalid responses will produce an error. A message will be displayed in the "Validation Summary" panel.

Invention Description

TOP

1. Invention Title *
Provide a fully-descriptive title for your invention.

Method to Reduce Huge Page Internal Fragmentation

2. Invention Description - Background *
Background: (a) Describe the problem solved by your invention by adding text only. Diagrams, drawings or files should be attached in Step 2. (b) List and briefly describe any existing products, publications, patents, and other works you are aware of that are most closely related to your invention. (c) If any of these works solve or address the same problem, what are the drawbacks of the known solution(s)?

As the price of DRAM memory goes down and demand for large memory capacity from memory intensive applications such as in-memory key-value store and NoSQL database goes up, more and more systems are now equipped with a large amount of memory, typically on the order of several hundreds of Gigabytes or even a few Terabytes.

With such large memory capacity, traditional use of 4KB page size puts a lot of stress on the virtual memory subsystem of the operating system (OS). This is because the huge number of 4KB pages required overwhelms the capacity of TLB. As a result, increased TLB misses incur high virtual address translation overhead and degrade system performance. The problem gets worse in a virtualized environment since both the guest and the host must perform virtual address translation.

Using a larger page size is an obvious and effective solution for the problem. In fact, modern CPUs (such as Intel x86, IBM Power, S390, etc.) and OSes (such as Linux, Mac OS, etc.) have supported larger page sizes (platform dependent) for quite some time now. And research [1] have shown that by simply enabling larger page sizes (typically called huge pages), some memory intensive application such as MongoDB can see as much as 37% performance improvement.

Despite its effectiveness, adoption of huge page (i.e., making huge page the default system page size) has been slow. The main reason is because of the application memory bloat due to fragmentation when using huge pages. There are two types of fragmentation: external and internal. We use the huge page size of 2MB on Intel x86 as an example to illustrate the two types of fragmentation.

External fragmentation refers to the situation where we cannot find 512 consecutive 4KB pages in order to map a 2MB page. As a result, even though the system might have plenty of free memory regions, all of them are smaller than 2MB so we cannot use 2MB pages to map them. Note that external fragmentation only exists when there are a mixture of different page sizes.

Internal fragmentation refers to the situation that the memory space within a page cannot be used by another application once the page is given to an application, despite the fact that the owner application might be using just one byte in the page. This is the natural side effect of using fix-sized pages as the smallest allocation unit. Note that internal fragmentation applies to both 4KB and 2MB pages. But it's obvious that the 2MB pages make the problem 512 times worse.

There are existing ways to deal with both types of fragmentation. For external fragmentation, the typical solution is to move 4KB pages around (i.e., remap them) in order to create 512 consecutive 4KB pages so that they can be mapped with one single 2MB page. For internal fragmentation, the typical solution is to break the 2MB page down into 512 4KB pages, and recycle those unused 4KB pages for other applications.

The internal fragmentation problem is fundamentally unsolved. Because the solution simply goes back to the original 4KB pages, which defeats the purpose and loses the benefits of using 2MB pages in the first place. In this invention, we present a method to really solve the internal fragmentation without falling back to the 4KB pages.

3. Invention Description - Summary *

Summary of Invention: (a) Briefly describe the main idea of your invention by adding text only (saving the details for Question 4 below). Diagrams, drawings or files should be attached in Step 2. (b) Describe the advantage(s) of using your invention instead of the known solutions (if any) described above, and how your invention is different compared to the known solutions.

The fundamental difficulty in dealing with internal fragmentation lies in the fact that an in-use memory chunk cannot be moved around (i.e., copied from one page to another) because the application holds a direct pointer to the chunk. So until the application calls free(), there is nothing the memory allocator can do to that chunk. Note that here we are talking about the C/C++ type of runtime environment where the applications must explicitly manage memory allocation through malloc()/free() calls. Runtime environment like Java JVM solves the problem by forbidding pointers and garbage collecting unused objects, and therefore are not the target of this invention.

In order to be able to move an in-use memory chunk, an obvious solution is that, instead of returning a direct pointer p to the chunk to the application, returning a pointer q that points to p to the application, i.e., q = &p using C/C++ notation. This way, the actual location of the chunk can be changed transparently to the application by modifying the value of p while keeping q constant during the lifetime of the chunk.

However, in order to make this work, two major challenges must be addressed.

First, because q is not a direct pointer to the memory chunk, the application must dereference it once before it can be used to address memory contents. That is, instead of writing p->foo, an application must write (*q)->foo. It would be impractical to require all applications to make such change. Therefore, there must be a way to perform the (*q) dereference without any application change. This invention solves the problem by employing compiler support. We augment C/C++ compiler with the function of tracking the pointer q returned by malloc() and automatically generate the (*q) dereference code before the first time it's used.

Second, in order to move an in-use memory chunk to a different location for the purpose of defragmentation, the memory allocator must make sure that the application is not actively using the chunk during the move. Therefore, there is a race condition between the application and the memory allocator. More specifically, the application may be accessing the chunk by reading the content of p (= *q), while the memory allocator may be changing the location of the chunk by modifying the content of p. This invention solves the problem by using a simple read-write lock. And once again, the application code that acquires/releases the read lock before/after reading the content of p is automatically generated by the compiler and requires no application change.

4. Invention Description - Details *

Details: Describe how your invention works and how you propose it be implemented. In addition, if you have any diagrams or flow charts or description files, please load them in the Documents section.

The detailed description of the invention is presented in the attached file in the Documents section.

Previous Inventions

5. Previous Invention Disclosures *

If you or your co-inventors are re-submitting this invention, please provide the invention disclosure number and give a brief explanation as to why you are re-submitting; OTHERWISE, indicate 'NA'.

NA

External Disclosure / Use / Commercialization

6. Inclusion in IBM Product or Service *

If your invention is included in a current or planned IBM product or service (including products and services that have been offered for sale), please provide a brief explanation (who, when, how) of such inclusion; OTHERWISE, indicate 'NA'

NA

7. Disclosure to Non-IBMers *

Aside from current or planned products or services, if have you otherwise disclosed, or is there a plan to disclose your invention to any non-IBMer, please provide the details of the disclosure (who, when, how, CDA number if known); OTHERWISE, indicate 'NA'.

NA

External Party Involvement

8. Activity with Customer *

If your invention was made in the course of any activity involving a customer (such as RFQ, IGS engagement), please describe the activity; OTHERWISE indicate 'NA'.

NA

9. Joint Development / Joint Study (other than Government Contract) *

If the invention was made in the course of an activity involving a non-IBM development partner (such as joint development or joint study activities), please describe the activity; OTHERWISE indicate 'NA'.

NA

10. Government Contract *

Was the invention made in the course of an activity involving a government or government agency?

No

11. Government Contract Number *

If the invention was made in the course of an activity involving a government or government agency, please provide the Government Contract Number. OTHERWISE indicate 'NA'.

NA

12. Government Agency *

Please select the Government Agency issuing the Prime Contract. You can do so by expanding the appropriate country and division. If this invention was NOT made in the course of an activity involving a government or government agency, select 'NA - Not a Government Contract'.

NA - Not a government contract

13. Project Name and/or Project Manager *

Please provide the project name and/or project manager's name (if known) or indicate 'NA'.

NA

14. Acquisition? *

If at least one inventor was working for the acquired company when the invention was conceived, please identify the acquired company; OTHERWISE indicate 'NA'.

NA

Standards and Use by Others

15. Applicable to a Standard? *

If the invention is applicable to an Information Technology (IT) standard such as those likely to be developed by organizations such as IETF, W3C, Oasis, ISO, IED or ITU, please indicate which standard and whether IBM is participating in development or usage of the standard; OTHERWISE indicate 'NA'.

NA

16. Use by Others *

What companies (other than IBM), open source projects, or industry standards are most likely to implement this invention?

OS distributors such as RedHat, SuSE, Debian, Ubuntu.

17. Discoverability of Use *

How easily can the use of the invention by a third party be detected/discovered?

3. With work, e.g., using test cases; but not reverse engineering

Invention Areas, Technologies

18. Functional Area *

810 Operating systems, highly available and fault-tolerent systems (Systems Area)

19. Technology Code *

179901 System and Algorithm Optimizations

20. Owning Division *

Yorktown Research Lab

21. Invention Disclosure Routing *

810 - Operating systems, highly available and fault-tolerent systems (Systems Area)

Export Control

TOP

NOTICE: It is your responsibility to correctly answer these export questions. Failure to do so can result in significant penalties. Either Question 22 is answered 'No' or you must select an option from Question 23 (displayed when Question 22 is answered 'Yes'. Prior to selecting an option from Question 23, please confirm your response with your manager. Review the guidance on the IBM Export Regulations Website

We are not accepting disclosures that fall under the ITAR classification. Please contact your IPLaw Department for assistance.

22. Non-Export Controlled Methods, Processes or Software *

Does this invention contain any export controlled content?

No

Examples of ITAR controlled content:

Source code, software, technology and related information pertaining to any of the following:

- Defense
- Military
- Government
- Intelligence
- Surveillance
- National Security
- Space (including commercial satellites)

Examples of Favored Nation Export Restricted content:

- Biometric software and technology for fingerprint and voiceprint recognition and analysis
- Source code and technology for Microprocessors (VHDL, netlists, etc), Servers (Power & Z architecture systems), High Performance Computing (DCS), and Integrated Circuits
- Magnetometry, Particle Physics, Nanotechnology, and Molecular Electronics technology
- Telecommunications technology (Digital RF, Software defined networks)
- Radiation and Temperature Hardened technology
- Neural/Neuromorphic Computing (e.g. TrueNorth), Quantum Computing

technology
- Nuclear Power technology

Examples of Encryption Restricted content:

Cryptographic or crypto-analysis component, system or enhancements thereto, including encryption/decryption algorithms or implementation