Requirements for Windows Thrashing Detection Tool

Joseph Caton
My Nguyen
Brooks Olney
Thomas Pannozzo
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620

Version 2.05
November 25, 2018

IMPORTANT NOTE TO THE STUDENT: The Introduction "sets the stage" for the requirements and precisely defines all key terms. You should *not* define well-known terms unless their meaning may be ambiguous. Any terms that are unusual or unique to your project *must be defined*. Sometimes a separate glossary may be necessary. Very often, a figure should be used in the opening paragraph (a picture is worth 10K words, etc.). A list of assumptions (or "givens") is important. The requirement items must be *numbered* and in order from most to least important. Every requirement item must be *measurable*. Footnotes are acceptable. Key standards (and/or existing work that is being built upon) must be cited and included in a List of References. Important details are the author(s) name and contact information, and the document name, version number, and date.

History of document

- **Version 2.05** (November 25, 2018) – M. Nguyen. Updated a user story.
- **Version 2.04** (September 28, 2018) – M. Nguyen. Removal of a user story.
- **Version 2.03** (September 21, 2018) – M. Nguyen. Updated the requirements to add User Stories and Acceptance Criteria.
- **Version 2.02** (September 13, 2018) – M. Nguyen. Updated the requirements based on customer feedback with User Stories and Acceptance Criteria format.
- **Version 2.01** (September 7, 2018) – M. Nguyen. Changed the requirements to SunView's Project Business Requirement Document with User Stories and Acceptance Criteria format.
- **Version 2.00** (August 24, 2018) – F. Giovannetti. Changed the requirements to User Stories format.
- **Version 1.03** (August 28, 2017) – Removed ambiguities in requirement #1 (added "by driving") and requirement #2 (added "per person")
- **Version 1.02** (January 8, 2014) – Corrected more typographical errors throughout the document
- **Version 1.01** (December 26, 2013) – Corrected typographical errors in the Introduction section
- **Version 1.00** (December 20, 2013) – Lunch requirements document created.

1. Introduction

The problem addressed by these requirements is, "When is the Windows OS thrashing and how can a detection tool be created to give an alert when thrashing occurs?" Prior to understanding thrashing, one must first understand the operating system, paging, and memory swapping. In an *operating system* (OS) within a virtual memory space, programs allocate memory from a physical address space to the processes within the programs (ref [1]). *Paging* is the rapid exchange of data in memory for data on a hard disk. A *page fault* happens when memory access requested does not map to something in RAM and *swapping* occurs when a page from RAM is swapped with a new page to be swapped back to RAM. *Thrashing,* also known as disk thrashing, happens when too many processes compete for a computer's virtual memory resources (ref [3]). From there, the computer becomes saturated due to the high memory usage, leading to a constant state of paging to the exclusion of most application-level processing (ref [2]). During thrashing, the *central processing unit* (CPU) spends more time swapping than running the processes. A *process* needs a sufficient number of memory pages to function properly. Thrashing can cause the performance of the computer to slow down and freeze the computer, preventing users to run applications or have active processes. Another sign of thrashing is when an application stops responding while the disk drive light blinks on and off (ref [3]). Some ways to resolve thrashing include increasing RAM, decreasing number of programs running on the computer, or adjusting the size of the swap file (ref [4]). Thrashing for the purposes of these requirements will be tested in a virtual machine by overloading memory and CPU with processes prior to integrating into SunView's machines.

2. Assumptions

The assumptions for this project are:

1. Thrashing will be measured in a virtual machine (VM).
2. Overload memory and CPU with processes to measure thrashing.

3. Requirements

User Stories:

1. As a developer, I want to measure the rate of change for page faults as it increases so that I detect it as a precursor of thrashing.
2. As a Developer, I would like to capture the system's high memory usage so that I can log it for further analysis of Windows OS thrashing.
3. As a Developer, I would like to capture the low CPU usage (due to every process waiting for the I/O to finish) so that I can log it for further analysis of Windows OS thrashing.
4. As a Developer, I would like to package these statistics in an organized manner so that I can format it into CSV and integrate it with an API.
5. As a Developer, I would like to force thrashing on my virtual machine so that I can train my thrashing detector model to automatically detect it.
6. As a Developer, I would like to implement machine learning in my product so that I can automatically determine the conditions of thrashing.
7. As an IT administrator, I need to know when one of my systems is thrashing, a sign of potential system performance issues, so that I can detect it and send a warning or alert.
8. As a developer, I want to limit the possibility of false positives as much as possible so that I can improve the accuracy of the tool.

Acceptance Criteria:

1. The application is successful in checking if thrashing is potentially occurring when a flag is raised for page faults.
2. The application will be considered successful when I know that the utilization of a system call returns the highest memory usage for each process identifier (PID).
3. The application will be considered successful when I know that the utilization of a system call returns the lowest CPU usage for each process identifier (PID).
4. The application will be considered successful when I know that the product successfully integrates with SunView's UI platform.
5. The application will be considered successful when I know that my detector model gives me a warning that a potential thrashing might occur.
6. The application will be considered successful when I know that my product gives me the conditions of thrashing.
7. The application will be considered successful when I know that my product identifies thrashing.
8. The application is successful when the alert has not been given until the minimum time period has elapsed.

Non-Functional requirements:

1. Detect which process causes thrashing.

List of References

[1] What is thrashing in an operating system?, Quora, 2018. URL:https://www.quora.com/What-is-thrashing-in-an-operating-system.

[2] Thrashing (computer science), Wikipedia, 2018. URL: https://en.wikipedia.org/wiki/Thrashing_(computer_science).

[3] Thrashing, Techopedia, 2018. URL: https://www.techopedia.com/definition/4766/thrashing.

[4] Thrashing, Computer Hope, 2017. URL: https://www.computerhope.com/jargon/t/thrash.htm.