Chalmers University of Technology and Gothenburg University

Operating Systems
EDA093, DIT 401

Exam 2024-08-20

Date, Time: Tuesday 2024/08/20, 14.00-18.00

Course Responsible:
Vincenzo Gulisano (031 772 61 47)

Auxiliary material: You may have with you
- An English-Swedish, Swedish-English dictionary.
- No other books, notes, calculators, etc.

Grade-scale ("Betygsgränser"):
CTH: 3:a 30-39 p, 4:a 40-49 p, 5:a 50-60 p
GU: Godkänd 30-49p, Väl godkänd 50-60 p

Exam review ("Granskningstid"):
Will be announced after the exam.

Instructions

- Do not forget to write your personal number, if you are a GU or CTH student and at which program ("linje").
- Start answering each assignment on a new page; number the pages and use only one side of each sheet of paper.
- Write in a **clear manner** and **motivate** (explain, justify) your answers. If it is not clear what is written, your answer will be considered wrong. If it is not explained/justified, even a correct answer will get **significantly** lower (possibly zero) marking.
- Please make an effort to have nice calligraphy!
- If you make **any assumptions** in answering any item, do not forget to clearly state what you assume.
- Answer questions in English, if possible. If you have a large difficulty with that and you think that your grade can be affected, feel free to write in Swedish.

Good luck !!!!

1. (4 p) What is the difference between task and data parallelism? Make a detailed example (NO pseudocode/code!) of an application that can leverage both task and data parallelism and explain how each technique would be used.

   [**Answer (in short):** In task, multiple threads are running different parts of a process. In data, multiple threads are running the same part on different portions of a process' data. Imagine a process that searches for mail addresses matching a certain regex from a long list, and then sends emails to such addresses. We could have 4 threads, two threads search one half of the list each (data parallelism), two threads send emails (data parallelism), and the first and second pair of threads perform different tasks in parallel (task parallelism). NOTICE: answering "an example could be an application in cloud computing that runs multiple tasks on shared data" is not a detailed example. Answering "task parallelism when tasks are parallel, data when data operations are parallel" is just rephrasing the question.]

2. (4 p) Suppose we have a multicore CPU server (with hundreds of cores) and we decide to host multiple VMs in it each running a data-intensive application (that is, continuously running heavy computations on data). Each VM gets a dedicated core and a dedicated portion of memory. What we observe is that, from the perspective of the applications, the performance does not exceed that of 10 VMs running in parallel, no matter how many cores/VMs we run. Would Amdhals' law explain why this is happening? Motivate your answer.

   [**Answer (in short):** No, it would not. Amdhals' law is about multithreaded processes and how they scale. Here we know nothing about the applications, not their serial/parallel portions. From what we know, the reason for a limited scalability could be a memory bottleneck at the hardware level. NOTICE: I have not asked you to write Amdhals' law nor to explain it. Answering yes or no without motivation is not answering.]

3. (4 p) Two friends meet to talk about their recent coding experiences. One says that, for a pool of threads, having as many threads as cores but not more is the best option. The other believes the number of threads should be way higher than the number of cores. Who do you think is right and why?

   [**Answer (in short):** They can be both right. If threads are running CPU-bound processes, once all physical cores (maybe hyper-threading ones too) are used, more threads will not lead to better performance. If threads are running IO-bound (or at least have higher degrees of IO than threads that are CPU-bound) and are blocked, a larger pool in this case can be beneficial.]

4. (4 p) Imagine a setup in which processes explicitly use physical addresses for their memory I/O. Draw and explain the workflow of checks performed before a process is given access to the main memory if base/limit registers are used.

   [**Answer (in short):** This question is asking about the Base and Limit register slide from the Memory Management slide.]

5. (4 p) Why do computers use RAM instead of just having the CPU perform I/O operations using the disk instead?

   [**Answer (in short):** The disk is slower than the main memory. Also, the CPU cannot directly read/write to disk usually but needs to go to a controller. Because of these two considerations, using disks instead of memory would imply way slower IO handling. EXTRA: similar reasoning can be made in connection to the use of main memory for instance for memory-mapped IO.]

6. (4 p) List and describe 3 different downsides/challenges of contiguous allocation for processes in main memory.

   [**Answer (in short):** 1. external fragmentation (sum of holes larger than the size of a process but no single hole large enough). 2. potentially fixed maximum size for a process once assigned a hole (if the previous/subsequent processes stay there, the process in between can only use the hole memory and not more) 3. potentially poor choice of a hole among several available ones (the algorithm used to chose a free hole might take a suboptimal decision). OTHERS: Long loading times since everything needs to be loaded. Long swapping times.]

7. (4 p) Why do sensitive instructions need to be a subset of privileged instructions to let VMs run code directly?

   [**Answer (in short):** A sensitive instruction does not necessarily trap if executed in user mode. A privileged does. If sensitive instructions are a subset of privileged, then they always trap, which means the guest VM will always trap when a sensitive instruction is run, give control to the hypervisor, and let the hypervisor take action accordingly.]

8. (4 p) How are Access Control Lists different from Capabilities?

   [**Answer (in short):** Access Control Lists are used to associate permissions (e.g., read, write, execute) on a per-file basis. Instead, capabilities associate permission to each process/domain.]

9. (4 p) How do stack canaries work?

   [**Answer (in short):** The idea is to put a special/random sequence of bytes (now known by the attacker) just after/below the return address and to check if such a value has been modified when returning from a function. If a buffer overflow attack modifies such value (that would most likely happen, although it is not necessarily the case) then the return address might also have been tampered with, and the effects of the attack can be prevented.]

10. (4 p) A process has 4 pages. None of the accesses to page 0 results in a page fault. Accesses to pages 1-3 result in page faults 50% of the time (for each page). How many distinct combinations of valid/invalid bits in the page table at the very beginning of the program execution could justify this behavior? Justify your answer.

    [**Answer (in short):** How the valid/invalid bits look at the very beginning of the execution does not necessarily imply anything on the number of page faults, since the page table can be further updated during the execution (e.g., with page replacements). From the question, we know that page 0 was loaded in the beginning, which leaves 2^3 combinations.]

11. (4 p) Can the optimal page replacement algorithm be implemented or not? Motivate your answer.

    [**Answer (in short):** No, because it would imply we know in advance in which order all pages will be accessed during the entire execution of a process, which is not a reasonable assumption for a generic process.]

12. (4 p) The LRU page replacement algorithm needs to maintain information about when a certain page has been accessed. What do you think is the information that a page replacement algorithm called Least Frequently Used should maintain instead?

[**Answer (in short):** If the last requires the time each page was last accessed (e.g., or some approximation of it), the least requires the number of times it was accessed (or some approximation of it), possibly over a given time period if such information is to be maintained e.g., only for a fixed/recent portion of time.]

13. (4 p) What is the difference between asynchronous and deferred target thread termination?

    [**Answer (in short):** Asynchronous terminates the thread immediately, deferred lets the target thread check if it should terminate and clean up its state before its termination.]

14. (4 p) What is the difference between a zombie and an orphan process?

    [**Answer (in short):** A zombie process is a process that has terminated but whose entry in the program table still exists (for instance a signal for a parent), while an orphan process is a process that has not terminated yet but its parent has.]

15. (4 p) Imagine that, in a producer/consumer setup, the consumer is faster in consuming an element than the producer is in producing one. What is better to favor a low memory consumption for the buffer used by the two, a bounded or an unbounded buffer? Explain your answer.

    [**Answer (in short):** Depends how one interprets "faster than the consumer": faster in absolute time but not necessarily once the execution starts, then a bounded buffer is safer (could happen for instance the consumer gets frequently preempted while the producer does not); faster as observed in execution, then it does not matter. In any case, a bounded buffer is always a safe choice.]