Principles of Computer Systems Design
Assignment 1

Tudor Dragan
Gabriel Carp

December 25, 2014

1 Question 1: Fundamental Abstractionst

A simple solution would be to lay out the machines in one single address space. If we assume that all the machines have the exact same memory size and they are perfectly fault tolerant, a simple arithmetic calculation can be used in the translation of the addresses for each machine. We could find the machine with *address* **mod** *machineMemorySize* and the find the local address by subtracting the *offSet*.

Because we are not working with an ideal scenario, we must translate the address by doing a look-up for finding the machine that contains the page[1] in memory. The look-up should be either a mapping or a function that retrieves the *machine number* and *local address* of the page. By using this solution, the memory can be spread out in different sized memory allocations on different machines.

In order to do these translations we would need to implement a centralized mapping system (a system similar to DNS for websites). In our case we use the **Central Server Algorithm** where a central-server maintains all the shared data. It translates the addresses and services the read requests from other nodes or clients by returning the data items to them. It updates the data on write requests by clients and returns acknowledgment messages. A timeout can be employed to resend the requests in case of failed acknowledgments. Duplicate write requests can be detected by associating sequence numbers with write requests. A failure condition is returned to the application trying to access shared data after several retransmissions without a response.

Although, the central-server algorithm is simple to implement, the central-server can become a *bottleneck*. To overcome this problem, shared data can be distributed among several servers. In such a case, clients must be able to locate the appropriate server for every data access. Multicasting data access requests is undesirable as it does not reduce the load at the servers compared to the central-server scheme. A better way to distribute data is to partition the shared data by address and use a mapping function to locate the appropriate server[2].

[1]A page, memory page, or virtual page is a fixed-length contiguous block of virtual memory, described by a single entry in the page table.
[2]`http://cs.gmu.edu/cne/modules/dsm/blue/ctr_ser_Al.html`

The API contains the address translations and the READ and WRITE functions that we assume as being atomic functions[3]. The READ and WRITE functions need to translate the address by calculating the address offset and machine identifier that has the page.

```
private {machineID, localOffset} TRANSLATE(address) {
        // the offset represents the first address of the machine
       {machineID, offset} = lookupMachine(address);
       // get the local offset in the machine's memory
       localOffset = address - offset;
       return {machineID,localOffset};
       }
```

Figure 1: Translate function

Essentially the translation gives us the machineID, to which we send the READ/WRITE request and the localOffset that provides the location where the machine stores the page. We use the lookupMachine function that returns us the machineID and the machine memory offset from the mapping of machines for the input address.

READ

The READ function can return a data object (for e.g. a memory page) or an error if the machine that we need to read from is unreachable.

```
private data/error READ(address) {
       {machineID, offset} = TRANSLATE(address);
       SEND (machineID, offset};
       //wait for the response; if TIMEOUT return error
       try RECEIVE data;
       catch TIMEOUT
              return error;
       return data;
       }
```

Figure 2: READ function

WRITE

The WRITE function keeps track of the data by handling a status response from the machine that we must write on to.

[3]In concurrent programming, an operation (or set of operations) is atomic, linearizable, indivisible or uninterruptible if it appears to the rest of the system to occur instantaneously. Atomicity is a guarantee of isolation from concurrent processes.

```
private status WRITE(address, data) {
       {machineID, offset} = TRANSLATE(address);
       SEND (machineID, offset, data};
       //wait for the response; if TIMEOUT return error
       try RECEIVE status;
       catch TIMEOUT
              return error;
       return success;
       }
```

Figure 3: WRITE function

We assumed that our distributed memory system would *always* have the machines in the system. The current name mapping does not allow dynamic joins and leaves without compromising memory segments. If a machine should fail the requested page upon a READ for example will be lost. To resolve this issue we should implement data migration, where every page is replicated on different machines. The look-up should be changed by extending the algorithm and should retrieve the cloned locations as well..

But what do we do about the machines that left address space? In order to implement dynamic joins and leaves we must manage several redundant machines. The central server monitors the status of each machine and modifies the mapping accordingly. The system should handle copying data to joining machines, to ensure true redundancy.

2 Question 2: Techniques for Performance

2.0.1 How does concurrency influence latency in a computer system? Is its influence always positive, always negative, or is there a trade-off?

Concurrency doesn't always guarantee a decrease in latency, it correlates to how the algorithm could be broken down into several tasks. If the processing parallelizes perfectly (i.e., each subtask can run without any coordination with other subtasks and each subtask requires the same amount of work),then this plan can,in principle,speed up the processing by a factor n,where n is the number of subtasks executing in parallel. In practice, the speedup is usually less than n because there is overhead in parallelizing a computation. Because the parallelization can't be perfectly balanced across n machines, we must wait for the slowest response in order to present the results.

Although parallelizing can improve performance, several challenges must be overcome. First, many applications are difficult to parallelize. And it still depends on the algorithm maybe by making it concurrent unexpected problems occur while separating tasks. Programmers must often struggle with threads and locks,or explicit message passing, to obtain concurrency. The trade-off is always an increase in development and testing time to reduce the latency.

2.0.2 Explain the difference between dallying and batching. Provide one example of each.

Dallying is the process of delaying requests in the chance that the initial request is overwritten by the following or that all the requests could be sent together in a process called batching. Batching is the process of sending several requests at a time which saves time instead of sending them individually. Both process are used to decrease latency.

Dallying also increases the opportunities for batching, thus we chose an example that implements both at the same time. When designing a network card, the packet buffer must be large enough to hold N packets and then only allow it to send an interrupt when it has received N packets. This way, the CPU won't have to check for data via polling, but it also won't get overwhelmed with interrupts because they will be occurring N times less often. This may increase the latency for an individual packet that is dallied but our overall throughput will increase since we are eliminating many of the expensive interrupts.

2.0.3 Is caching an example of a fast path optimization? Explain why or why not.

Caching is the most common example of optimizing for the most frequent cases. For example, most Web browsers maintain a cache of recently accessed Web pages.This cache is indexed by the name of theWeb page and returns the page for that name.If the user asks to view the same page again, then the cache can return the cached copy of the page immediately (a fast path); only the first access requires a trip to the service (a slow path). Of course we always need to verify the content of the web page, if it has been modified or not. The only overhead here is checking the freshness and validity of the page.

Programming Task

rateBooks

In the **rateBooks** method in the **CertainBookStore** adds a ratings to the given set only if all the items in the set are valid. If one does not meet the criteria we don't rate any of them and throw an exception.

getTopRatedBooks

The **getTopRatedBooks** method we needed to retrieve the books sorted by a rating. This is simply done by implementing a custom class that implements the *Comparator* interface and applying it to the *Collection.sort* method. If the user requests **k** books and **k** is larger than the number of books in the store, we simply retrieve the entire list of books and do not throw an exception.

getBooksInDemand

Returns the books that have the amount of sale misses larger than 0.

Discussion on architecture

The architecture is strongly modular by using good design patterns and enforcing an approach based on interface implementation. All the classes are clearly defined and don't overlap and changing one doesn't impact other parts of the code. All in all this design uses the core principles of OOP and ensures scalability as well as robustness. We can easily add or remove proxies without compromising the server-side.

The architecture of the solution takes into account the matter of isolation by differentiating the calls from each client to a different address. And it protects these calls by serializing the sensitive data sent from the stock manager. While both do some basic data verification before actually sending the request to the server.

The architecture does not implement a naming service because the service is accessible locally. However, we do specify which IP and port the proxies connect to. If we would use a naming system like DNS for example, we could access the servers by simply accessing a link like *www.bookstore.com.*

Search Path is a naming mechanism that allows clients to discover and communicate with services, not by using it's own AI to become the new SkyNet while finding and dealing with services but by simply going though a list of possible addresses that could respond and passing each of them in order till they respond as expected.