Don't be afraid of the mainframe

A demystifying guide

The layman's introduction to IBM mainframes

Niek de Greef

Table of contents

Introduction

This book is an introduction to IBM mainframes. Mainframe computing is different and often not well understood. This book aims to describe IBM mainframe technology in easily understandable terms. The reader is expected to have no more than a basic understanding of computers. My aim with this book is to clear the smoke and mirrors around mainframes and the use of mainframe technology.

In this book, I will not dive deep into technology. However, technical descriptions are necessary here and there to explain certain concepts. It is okay if you skip the parts that go too deep.

The book's title is Don't Be Afraid of The Mainframe because that is what I was aiming to achieve in writing this book: to make the reader more comfortable with IBM mainframes. In my experience, we, as mainframers, are often very good at scaring non-mainframers with weird technical jargon. Or, to put it more lightly, we are not very good at explaining mainframe things in layperson's terms. The mainframe is as simple (or complex) as any other computing tool. Perhaps it is even simpler.

I will discuss technical aspects around IBM mainframes and commercial and financial considerations. These aspects are related to each other, as I will explain. Some basic concepts must be understood even for a person only interested in the financial aspects of IBM mainframe technology. I will do my best in this book to address this challenge.

I hope you enjoy reading the book. All feedback is welcomed. Please send me an email at: …

Niek

1 What is a mainframe?

Before anything else, let's first define what a mainframe is. After that, I will give a very brief historical perspective on mainframes.

The mainframe: a definition

I will make up my definition based on definitions from several other resources.

A mainframe is a large central computer designed to run many different computing tasks simultaneously in a secure manner.

In this definition, I use the term computer for all the hardware and software components needed to run a meaningful application software.

There are a few critical aspects in this definition of a mainframe.

Central. The design point of the mainframe is to provide a centralized function for many computing tasks. The mainframe shares computing resources, hardware, and software between applications. The other way around, many applications use the same mainframe computer.

Many tasks at the same time. From the ground up, the mainframe operating system design is meant to run many technical tasks and applications very efficiently. The operating system is the core software on a computer on which application software can be executed.

Secure. Because many applications run on the same computing platform, the mainframe's operating system has very sophisticated facilities to ensure that applications can not interfere with each other.

A very brief history

Mainframes stem from a time when computing equipment was costly, and a single central solution for computing was the only way to run computing tasks economically.

Many large organizations use mainframes. You typically find mainframes in banks, insurance companies, government institutions, large supermarkets, and many smaller organizations. These organizations have often used the mainframe since the 1960s or 1970s. Applications that were developed then are often still in use today.

Typical programming languages used to build business applications for the mainframe are COBOL and PL/I, although Java has also become a very popular language on the mainframe.

Other modern technologies and middleware are nowadays also available on the mainframe. You can run state-of-the-art application servers, integration tools, and data analytics solutions and even run Artificial Intelligence applications with special AI facilities on a mainframe.

A mainframe is broadly appreciated for its ability to process and manage large amounts of data very robustly and securely.

Many companies in the past 70 years have built mainframe computers. Examples are ICL, Bull, Sperry, Tandem, Siemens, Unisys, and NEC. However, IBM has been by far the most successful mainframe manufacturer, and IBM mainframes have become synonymous with the term 'mainframe.' In this book, I will focus on the IBM mainframe solely.

2 Understanding mainframe cost

Let's first talk about money. The main thing that scares mainframe users is the bill they get from their software and hardware suppliers. This chapter will describe the most important costs of running a mainframe. Then, I will explore the good, the bad, and the ugly of mainframe costs.

An overview of the cost of a mainframe

Mainframe costs fall into three categories: hardware, software, and management.

- Management costs are the costs to install, maintain, and operate the hardware and software in a mainframe configuration.
- Hardware costs include the acquisition and maintenance costs of the mainframe server machines(s), the storage devices, and all the other equipment you need.
- Software costs are the licenses paid for the required software components.

The most significant cost component of a mainframe setup is the software. I estimate unscientifically, but from experience, that two-thirds of the total mainframe costs are software costs. Hardware and management costs are half of the remaining costs, so each is about one-sixth of the entire sum.

The cost of managing a mainframe configuration

The costs of managing the mainframe are people costs. I will describe what kind of people you need and elaborate on the associated costs for these roles.

Who do you need to run a mainframe?

To run a mainframe computer requires a few roles. Roles will be combined into one or more persons for small mainframe shops. In larger mainframe setups, you will find that roles are even more specialized. The terminology I have used here is mine. Role names will differ across organizations, but the responsibilities are the same everywhere.

- Engineering. The engineering team installs and configures mainframe hardware, operating system, and middleware software components.
- Operations. A centralized operation team ensures the daily operation and functioning of the central components of the mainframe configuration.
- Application management. The application management team manages the daily functioning of one or more applications running on the mainframe. They perform this task as part of a business process that the application serves. In a DevOps organization, these teams perform the Ops side of a DevOps team.

5 z/OS – the mainframe flagship operating system

This chapter will introduce you to the main z/OS concepts and terminology. The goal is to give you a good idea about the main z/OS concepts. I will also explain how z/OS peculiarities relate to more commonly known concepts in Unix and Windows operating systems.

I will discuss z/OS from the traditional 'MVS' side and the more relatable Unix side. The traditional MVS side stems from the 1960s and deviates most from what we know from Windows and Unix. The second part is z/OS Unix, an extension added in the 1990s, and can run Unix applications, very much like other Unix flavors.

In the section following I will talk about the unique clustering facility that z/OS brings, called parallel sysplex.

Finally, I will look at the green screen user interface that the mainframe is often associated with. I will discuss where the tools with green screen are still necessary and describe the modern tools for z/OS, tools with more sophisticated graphical user interfaces.

The MVS part of z/OS

The part of z/OS that I refer to as the MVS side, is the traditional 'old-school' mainframe side. The MVS part is the subset of functions rooted in the 1960s. These functions were built in an era in which punch cards contained the program code. You fed a stack of punch cards into a punch card reader connected to the computer to get something done from the computer. In the MVS part of z/OS we still find the remains of these features. Today these look a bit awkward.

Basic operation of the MVS part with batch and JCL

First, I will look at the basic operation of a batch process in z/OS. Batch processing is the style of computing with which you prepare input data to be processed, start the compute process, wait a while, and receive the output data as a result. There is no interactivity in a batch process.

The batch process is a core concept in the MVS part of z/OS. In essence, today's batch processing still works in the same way as it was when designed in the 1960.

To run a program on z/OS, you need a mechanism to tell z/OS what to do. You do this by creating a script. The language for this z/OS script is called JCL – Job Control Language.

With the JCL script you tell the computer which program to run, what the input is for that program and where to write the output. The JCL code looks like this:

```
//RUNPROG   EXEC PGM=PROGRAM1
//INPUT     DD DISP=SHR,DSN=MY.INPUT.DATA
//OUTPUT    DD DISP=NEW,MY.OUTPUT.DATA
```

This code looks awful of course, despite its simplicity. The JCL language uses strange forward slashes, fixed positions of key fields, etcetera. Many of these characters exist because the JCL

language was designed for punch cards. Punch cards could carry maximum 80 characters per line, and every line also needed some particular positions to control the operation of the punch card reader. Everything needed to be coded in uppercase to make things as simple as possible for the punch card reader. JCL is easily readable for a punch card reader device, but from an aesthetical and ergonomic perspective, it is a quite horrendous phenomenon. Punch cards have disappeared, but the JCL language still exists.

Returning to the above JCL snippet, you tell the computer to run PROGRAM1 with this script, use label INPUT as input file, referring to file MY.INPUT.DATA, and label OUTPUT as output file, referring to file MY.OUTPUT.DATA. If you feed this into the mainframe running z/OS it will try to execute this as such. Figure 8 illustrates this process.

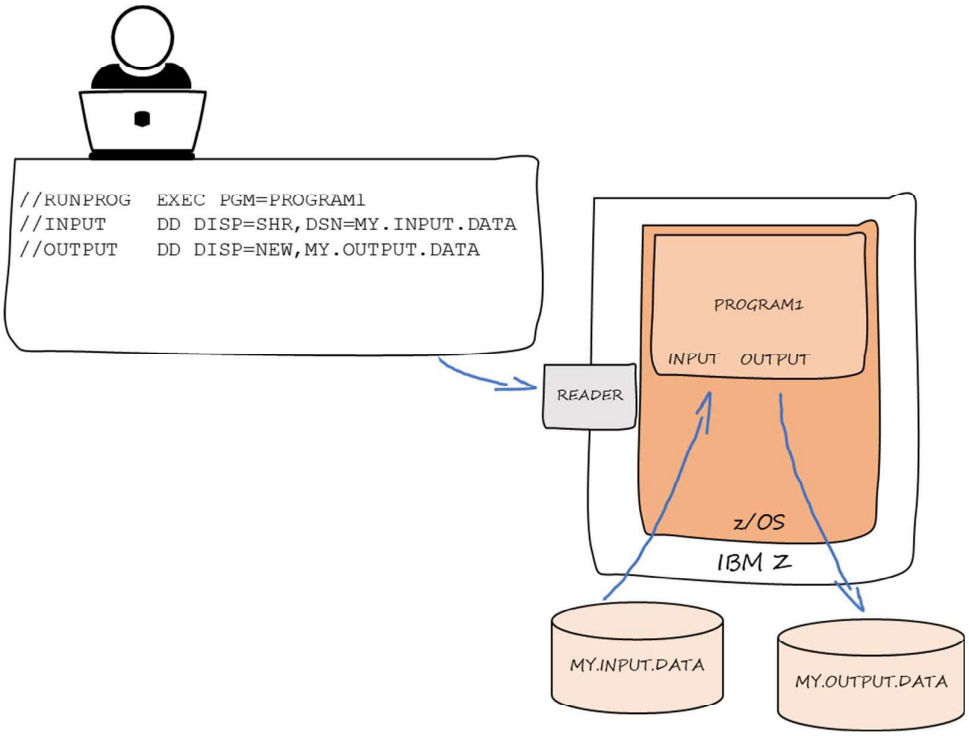

Figure 8 JCL to describe work to be done

In the old days the programmer punched the JCL on paper punch cards with a special device, a punch card machine, creating one card for every line of JCL. She then carried the stack of cards to the building where the computer was installed, and inserted into a punch card reader attached to the mainframe. The punch card reader read the stack of cards and executed the program.

Nowadays card reader devices do not exist anymore. They are replaced by a piece of software in z/OS. This software function is called the internal reader.