Boot Loader

구조설계서

2022-05-01

홍길동

이 문서는 소프트웨어 구조 설계자 양성 프로그램에서 양성 인력이 작성한 **Boot Loader** 개발을 위한 구조 설계서입니다.

(주) 보이는 소프트웨어 연구소
조용진(drajin.cho@bosornd.com)

REVISION HISTORY

Version	Date	Author	Description
0.1	2022-04-04	홍길동	초기 문서 생성
0.2	2022-04-11	홍길동	품질 시나리오 Revision
0.3	2022-04-17	홍길동	후보 구조 분석 추가
0.4	2022-04-25	홍길동	최종 구조 설계 Revision
0.5	2022-05-01	홍길동	시스템 구조 기술

1. 시스템 정의

1.1. 시스템 개요

Boot Loader (Boot Loader)란 운영체제가 시동되기 이전에 미리 실행되면서 커널이 올바르게 시동되기 위해 필요한 모든 관련 작업을 마무리하고 최종적으로 운영체제를 구동시키기 위한 목적을 가진 프로그램이다.

아래 그림 1은 부트 시퀀스를 나타낸다. 컴퓨터 전원이 켜지면, ROM (Read-only memory)에 들어있는 BIOS가 로드 된다. BIOS는 컴퓨터에 연결된 저장 매체에서 설정된 부팅 순서대로 Boot Loader를 불러오게 된다. 이 때, 하드 디스크가 첫 번째 부팅 장치로 설정되어 있으면, BIOS는 하드 디스크의 MBR (Ma_ster Boot Record) 영역에 저장된 Boot Loader를 로드한다. Boot Loader는 커널을 시동시키기 전 필요한 하드웨어들을 준비하고, 커널 이미지를 메모리로 로드 한다.

그림 1. 부트 시퀀스.

본 문서에서는 데이터센터 서버에 설치되는 Boot Loader의 소프트웨어 구성 항목의 구조적 설계를 다룬다. 아래 그림 2는 데이터센터에서 널리 사용되는 서버 구조를 나타낸다. 데이터센터용 서버에는 많은 수의 CPU 코어 (약 50개 ~ 100개), DRAM, GPU, NIC, HDD/SSD 등 다양한 컴포넌트들이 장착되어 있다. 주로 Windows나 Linux가 설치되면, 커널 수행을 위해 필요한 동작들이 많다. 따라서 Embedded 환경과 비교하여 BIOS와f Boot Loader도 다양한 기능을 제공하고 있다.

(주) 보이는 소프트웨어 연구소
조용진(drajin.cho@bosornd.com)

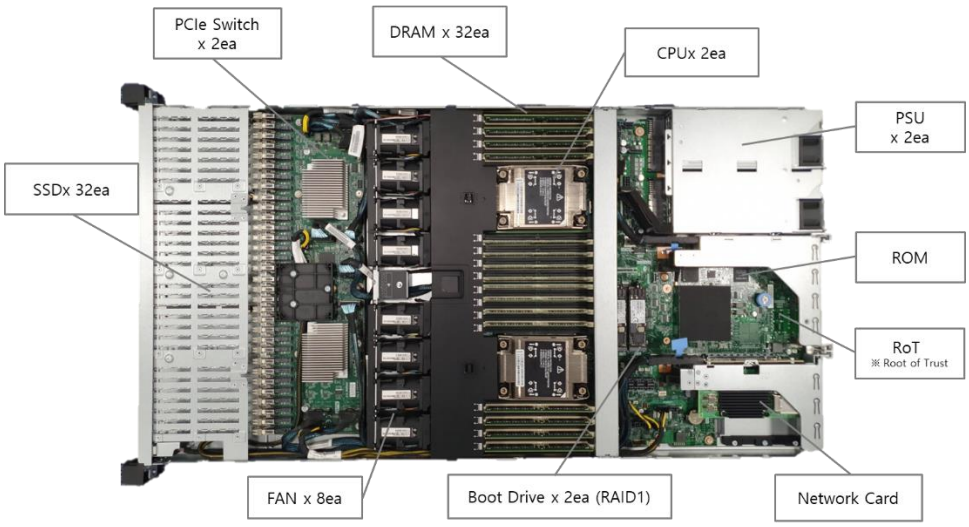

그림 2. 서버 구조.

아래 그림 3은 데이터센터 서버에서 사용되는 Boot Loader 컴포넌트 구성 및 역할을 나타낸다. Boot Loader는 Storage의 MBR (Ma_ster Boot Record)영역에 존재하며, 파일 시스템 상에 존재하는 Kernel Image를 DRAM으로 로드 해주는 역할을 담당한다. 이를 위해, File System 및 필요한 Device Driver를 포함하고 있다. 추가로, 사용자에게 시스템 점검 및 부팅 환경 설정을 위한 Shell 기능도 제공한다.

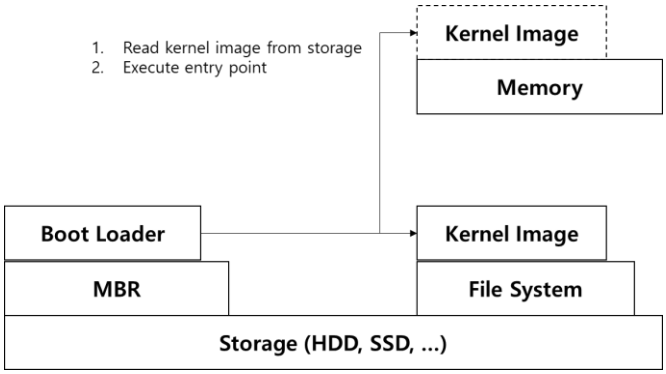

그림 3. Boot Loader 동작.

1.2. 시스템 정의

본 문서에서 정의하는 Boot Loader 개요 및 개발 범위는 그림 4와 같다.

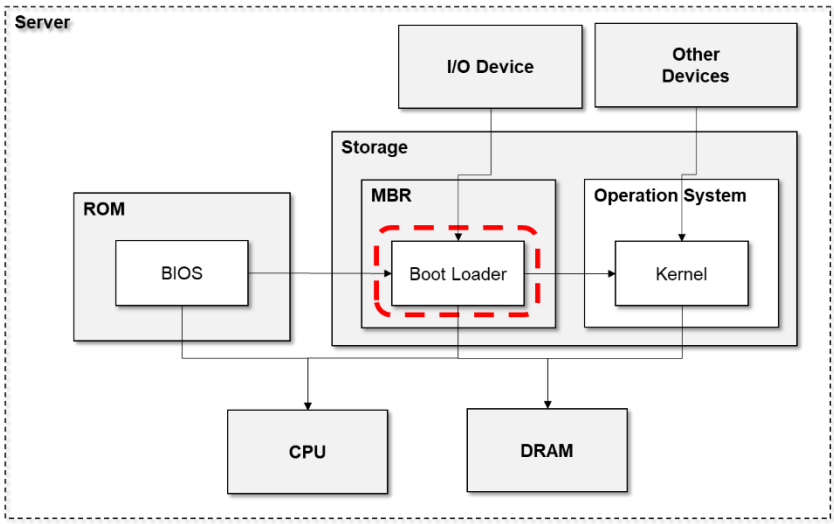

그림 4. 시스템 Layer 및 개발 범위.

1.2.1. BIOS

BIOS는 서버 전원이 켜지면 CPU가 처음으로 수행하는 소프트웨어이다. CPU, DRAM, Disk에 대한 POST (Power-on-Self-Test)를 수행한 후, MBR에 저장된 Boot Loader를 DRAM에 로드한 후, 이를 수행한다.

1.2.2. User

User는 서버 사용자이며, Boot Loader Configuration 설정, Kernel Image를 제공하며, Shell Command를 통해 Boot Loader가 제공하는 명령어를 입력한다.

1.2.3. Kernel

Kernel은 운영 체제의 핵심이 되는 소프트웨어로 스토리지에 저장되어 있다가, Boot Loader로부터 수행된다. Kernel은 (1) 메모리 관리, (2) 프로세스 관리, (3) 장치 드라이버 관리, (4) 시스템 호출 및 보안을 담당한다.

1.2.4. DRAM

휘발성 메모리로 Kernel Image 수행을 위해 로드하는 용도로 사용된다.

1.2.5. I/O Device

Keynote, 모니터 등 사용자와의 입출력을 위한 주변 장치를 의미한다.

1.2.6. Storage

비휘발성 메모리이다. Boot Loader나 Kernel Image를 수행하기 위한 공간으로 사용한다.

1.2.7. 시스템 경계

Boot Loader는 BIOS로부터 DRAM에 로드 되어, 스토리지에 저장된 Kernel Image를 DRAM으로 로드 하고, 이를 호출한다. 또한, Boot Loader는 사용자로부터 I/O Device를 통해 필요한 설정 값 및 명령어를 입력 받아, 이를 처리하고, 그 결과를 보여준다.

따라서 본 문서에서 설계하고자 하는 Boot Loader 경계는 아래 그림과 같다.

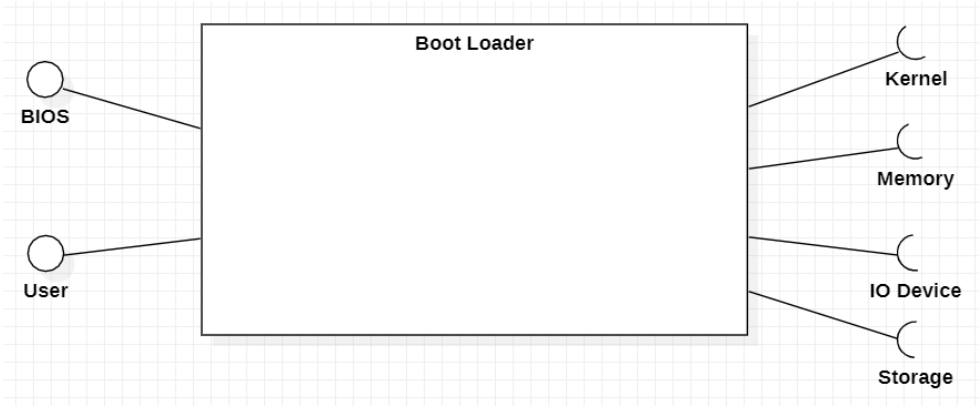

그림 5. 시스템 경계.

1.3. 시스템 제약 사항

- 서버용 보드에 설치된 BIOS는 Embedded용 BIOS와 달리 CPU, DRAM, Disk에 대한 POST (Power-on-Self-Test)를 수행한다. 따라서 Boot Loader에서는 Kernel Image 로드 성능을 위해 추가적인 POST를 수행하지 않는다.
- Kernel Image는 파일 시스템의 파일 형태로 저장된다. Ext2, Ext3, Ext4, XFS를 지원하여야 하며, 파일 시스템 UUID로 판별하여야 한다.
- 서버에 장착된 IO Device는 Serial Port 장치로 제한한다. 데이터센터의 특성 상 사용자가 직접 서버에 접근하는 경우가 매우 드물며, 따라서 가장 간단한 형태의 IO Device만 장착된다.

1.4. 비즈니스 드라이버

Boot Loader 설계를 위해 고려해야할 주요 비즈니스 드라이버는 아래와 같다. 비즈니스 드라이버는, 본 과제의 구조설계에 있어서 중요한 품질이 무엇인가를 판단하는데 중요한 척도로 사용된다.

- Boot Loader는 최대한 빨리 부팅 작업을 완료하여, 사용자의 만족도를 높여야 한다.
- 개발할 Boot Loader는 디바이스 장치 추가 및 변경에 대해 신속하게 대응하여 원하는 기간 내에 제품 개발을 완료할 수 있어야 한다.
- Booting 과정에서 발생하는 장애 상황으로 인한 사용자의 불편은 최소화되어야 한다.
- 일부 사용자의 악의적인 공격에 대한 대비가 충분히 되어있어야 한다.

2. 요구사항

2.1. 기능적 요구사항

본 시스템의 Use Case Diagram은 아래 그림과 같다.

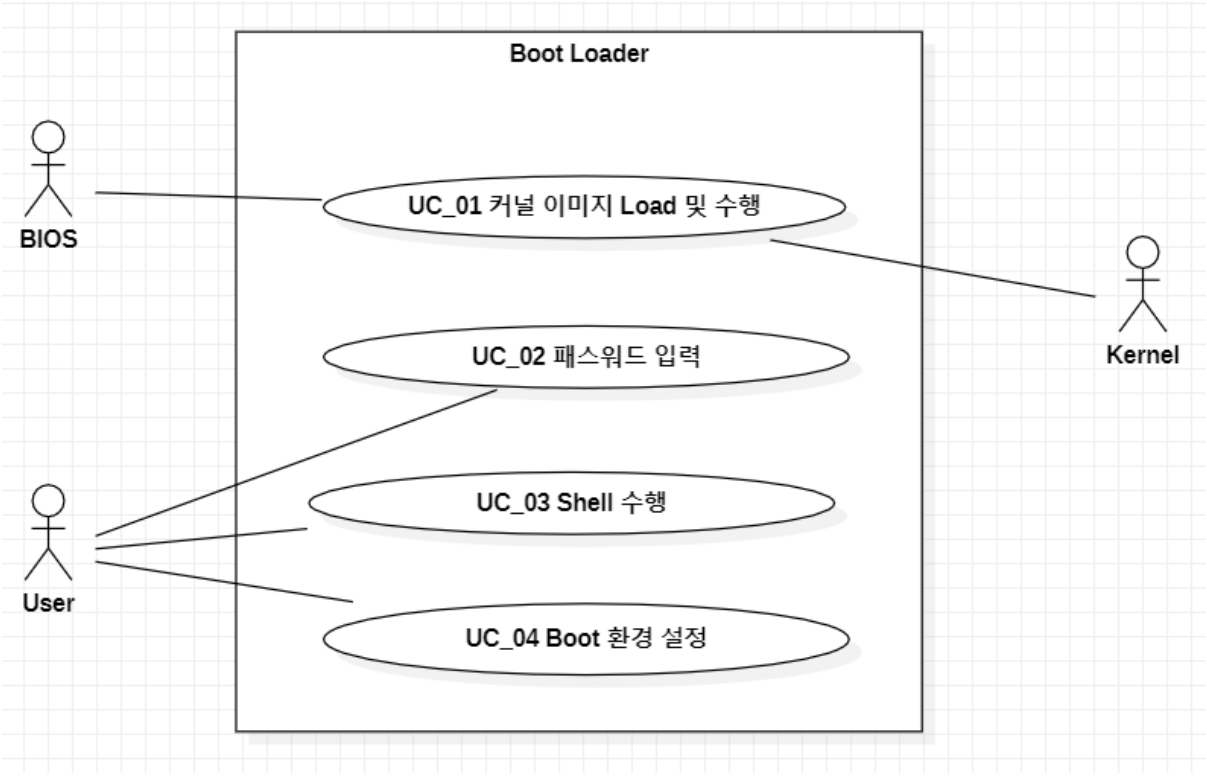

그림 6. Use Case Diagram.

2.1.1. UC_01 커널 이미지 Load 및 수행

UC_01	커널 이미지 Load 및 수행
설명	Boot Loader는 Booting Device에 저장된 커널 이미지를 DRAM에 로드하고, 수행한다.
행위자	BIOS
선행조건	Boot Loader 로드를 성공적으로 완료한 상태
후행조건	커널 이미지가 실행된다.

기본 동작	1. BIOS 는 Boot Loader 를 수행한다. 2. Boot Loader 는 Boot Config 파일로부터 수행에 필요한 설정 값을 세팅한다. 3. Boot Loader 는 파일 시스템에서 커널 이미지 파일을 읽어와 DRAM 에 로드 한다. 4. Boot Loader 는 커널 이미지 Signature 확인을 통해 커널 이미지 변조 여부를 체크한다. 5. Boot Loader 는 Shell Mode 진입 대기 시간 (10s) 동안 대기한다. 6. Boot Loader 는 커널 이미지를 수행한다.
추가 동작	AF1. 사용자 패스워드 입력 분기점: 기본 동작 2 1. 사용자 패스워드가 설정되어 있을 경우, Boot Loader 는 'UC_02 패스워드 입력'을 우선 수행한 후, 기본 동작 2 로 돌아간다.
	AF2. Boot Config 파일 미존재 분기점: 기본 동작 2 1. 미리 생성한 Boot Config 파일이 존재하지 않을 경우, Boot Loader 는 Default 설정 값을 읽어온 후, 기본 동작 3 으로 돌아간다.
	AF3. Kernel Image 미존재 분기점: 기본 동작 3 1. 만약 Kernel Image 가 존재하지 않는다면, Boot Loader 는 "Kernel 이 존재하지 않습니다."와 같은 적절한 에러 메시지를 출력한 후, 본 Use Case 를 종료한다.
	AF4. 커널 이미지 변조 발생 분기점: 기본 동작 4 1. Kernel Image 가 변조되었다면, Boot Loader 는 "Kernel Image 가 올바르지 않습니다."와 같은 적절한 에러 메시지를 출력한 후, 본 Use Case 를 종료한다.
	AF5. Shell Mode 진입 분기점: 기본 동작 5 1. 대기 시간 동안 User 가 Shell Mode 진입 키 (ex. F12)가 입력하였다면, "UC_03 Shell 수행'을 수행한 후, 본 Use Case 를 종료한다.

2.1.2. UC_02 패스워드 입력

UC_02	패스워드 입력
설명	User는 보안성 _향_상_을 위해, Boot Loader 수행 패스워드를 설정할 수 있다.
행위자	User
선행조건	- 'UC_01 Boot Loader 수행' 2단계까지 수행된 상태 - 사용자 패스워드가 설정된 상태
후행조건	'UC_01 Boot Loader 수행' 3단계부터 수행 재개된다.
기본 동작	1. Boot Loader 는 User 가 패스워드를 입력할 때까지 대기한다. 2. User 는 미리 설정한 패스워드를 입력한다. 3. Boot Loader 는 User 가 기설정한 패스워드를 MBR 로부터 읽어온다. 4. Boot Loader 는 읽어온 패스워드를 복호화 한다. 5. Boot Loader 는 User 가 입력한 패스워드와 기설정한 패스워드가 일치하는지를 체크한 후, 일치한다면 본 Use Case 를 종료한다.
추가 동작	AF1. 잘못된 패스워드 입력 분기점: 기본 동작 3 1. User 가 잘못된 패스워드를 입력할 경우, Boot Loader 는 "잘못된 패스워드입니다."와 같은 적절한 에러 메시지를 출력한 후, 1 단계를 수행한다.
	AF2. 잘못된 패스워드 입력 분기점: 기본 동작 3 1. User 가 5 회 이상 잘못된 패스워드를 입력할 경우, Boot Loader 는 "잘못된 패스워드입니다. 더 이상 부팅을 진행할 수 없습니다."와 같은 적절한 에러 메시지를 출력한 후, 부팅 과정을 종료한다.

2.1.3. UC_03 Shell 수행

UC_03	Shell 수행
설명	User는 시스템 점검 및 부팅 환경 설정을 위해 Shell을 수행한다.
행위자	User
선행조건	'UC_01. 커널 이미지 Load 및 수행' 2단계까지 수행된 상태
후행조건	
기본 동작	1. User 는 Boot Loader 수행 과정에서 Shell Mode 진입 키 (ex. F12)를

	입력한다. 2. Boot Loader 는 Shell Mode 수행한다. 3. Boot Loader 는 User 가 명령어를 입력할 때까지 대기한다. 4. User 는 미리 정의된 명령어 (아래 참조)를 입력한다. **[부팅 관련]** - root: 입력한 장치를 root device 로 지정 - blocklist: 입력한 파일이 저장된 blocklist 를 확인 - kernel: 부팅에 사용할 커널 이미지 파일 경로를 지정 - boot: 지정된 커널로 부팅 - reboot: 시스템을 재부팅 시키는 명령어 - halt: 시스템을 정지시키는 명령어 **[파일 관련]** - configfile: 지정 한 파일로부터 설정을 로드하는 명령어 - cat: 지정한 파일 내용 확인 - find: 지정한 파일이 위치한 장치명을 찾아주는 명령어 **[장치 관련]** - displayapm: APM (Advanced Power Management) BIOS 정보 출력 - displaymem: 물리적으로 DRAM 이 설치되어 있는 시스템 주소 공간에 대한 map 표시 - geometry: 지정된 device 에 대한 정보 출력 **[기타]** - help: Shell 명령어들에 대한 도움말 출력 - clear: 화면을 지우는 명령어 - md5crypt: Boot Loader 패스워드 설정을 위한 MD5 암호 문자 생성기 - quit: Shell Mode 종료 커맨드 5. Boot Loader 는 User 가 입력한 커맨드의 Valid 여부를 검증한다. 6. Boot Loader 는 커맨드를 처리한다. 각 명령어별 상세 처리 과정은 아래와 같다. - 부팅 관련 명령: Boot Loader 는 커널 이미지가 저장된 hard disk 를 access 하여, 이를 DRAM 으로 로드한 후, 커널에 제어권을 넘긴다. - 파일 관련 명령: Boot Loader 는 파일 시스템으로부터 User 가 지정한 파일을 읽은 후, 내용을 로드하거나, 출력한다. 혹은 Boot Loader 는 User 가 지정한 파일의 위치를 출력한다. - 장치 관련: Boot Loader 는 User 가 원하는 장치로부터 정보를 읽어와 이를 출력한다.

	- 기타: Boot Loader 는 명령어들에 대한 상세 도움말을 출력하거나, Shell 화면을 지우는 등 User 가 입력한 커맨드를 수행한다. 7. Boot Loader 는 커맨드 처리 결과를 출력한다. 8. 본 Use Case 의 3 단계를 수행한다.
추가 동작	AF1. Invalid 커맨드 입력 분기점: 기본 동작 5 1. User 가 입력한 명령어가 Valid 하지 않다면, Boot Loader 는 "잘못된 명령어입니다.", "Parameter 가 올바르지 않습니다." 등과 같은 적절한 에러 메시지를 출력한 후, 본 Use Case 의 3 단계를 수행한다.
	AF2. Quit 명령어 입력 분기점: 기본 동작 6 1. User 가 종료 명령어 (ex. quit)을 입력하였다면, Boot Loader 는 Shell Mode 를 종료하고, 본 Use Case 를 종료한다.

2.1.4. UC_04 Boot 환경 설정

UC_04	**Boot 환경 설정**
설명	User는 Boot Loader 수행에 필요한 설정 값을 변경할 수 있다.
행위자	User
선행조건	설정 값의 위치가 이미 정의된 상태
후행조건	User 설정 값으로 'UC_01 커널 이미지 Load 및 수행' 3단계를 수행한다.
기본 동작	1. User 는 미리 정의된 위치에 Config 파일 (ex. /boot/bl/bl.conf)을 생성한다. 2. Boot Loader 는 Config 파일로부터 설정 값을 읽어온다. 3. Boot Loader 는 설정 값의 Valid 여부를 체크한다. 4. 설정 값이 Valid 할 경우, Boot Loader 는 해당 설정 값으로 내부 설정 값을 업데이트 한다.
추가 동작	AF1. Invalid 설정 값 분기점: 기본 동작 3 1. 설정 값이 Valid 하지 않을 경우, Boot Loader 는 "Configuration File 에 Invalid 한 설정 값이 기술되어 있습니다."와 같은 적절한 에러 메시지를 출력한 후, 본 Use Case 를 종료한다.

2.2. 비기능적 요구사항

2.2.1. NFR_01 Boot Loader 수행 시간

NFR_01	Performance	Boot Loader 수행 시간
설명	Boot Loader는 10초 내에 Kernel 부팅에 필요한 모든 동작을 완료하여야 한다.	
환경	BIOS가 정상적으로 동작하는 상태	
자극	BIOS 가 Boot Loader 를 수행한다.	
반응	1. BIOS 는 Boot Loader 를 수행한다. 2. Boot Loader 는 수행에 필요한 설정 값을 세팅한다. 3. Boot Loader 는 파일 시스템에서 커널 이미지 파일을 읽어와 DRAM 에 로드 한다. 4. Boot Loader 는 커널 이미지 Signature 확인을 통해 커널 이미지 변조 여부를 체크한다 5. Boot Loader 는 커널 이미지를 수행한다.	
측정	[부팅 시간] = [Kernel 수행 시각] – [Boot Loader 수행 시작 시각]	
제약	[부팅 시간] <= 10s	

2.3. 품질 속성

2.3.1. QA_01 Boot Loader 수행 시간

QA_01	Performance	Boot Loader 수행 시간
설명	Boot Loader의 수행 시간은 빠를수록 좋다.	
환경	BIOS가 정상적으로 동작하는 상태	
자극	BIOS 가 Boot Loader 를 수행한다.	
반응	1. BIOS 는 Boot Loader 를 수행한다. 2. Boot Loader 는 수행에 필요한 설정 값을 세팅한다. 3. Boot Loader 는 파일 시스템에서 커널 이미지 파일을 읽어와 DRAM 에 로드 한다. 4. Boot Loader 는 커널 이미지 Signature 확인을 통해 커널 이미지 변조 여부를 체크한다 5. Boot Loader 는 커널 이미지를 수행한다.	

측정	[부팅 시간] = [Kernel 수행 시각 (반응5)] – [Boot Loader 수행 시작 시각 (반응 1)]

2.3.2. QA_02 디바이스 추가 및 변경 용이성

QA_02	Modifiability	디바이스 추가 및 변경 용이성
설명	Storage, IO Device의 다양한 디바이스가 추가되거나, 이미 장착된 디바이스 드라이버가 변경되었을 때, 시스템 변경을 위한 개발 비용이 최소화되어야 한다.	
환경	개발 완료 이후 시점 혹은 해당 요구사항 구현 진행 중인 시점	
자극	디바이스 추가 및 변경에 대한 신규 요구사항 요청	
반응	요구사항을 만족하도록 Boot Loader 를 수정 개발한다.	
측정	[개발 비용 (M/M)] = [수정, 검증을 다시 해야 하는 모듈] / [파일의 크기(LoC)]	

2.3.3. QA_03 Shell Command 처리 성능

QA_03	Performance	Shell Command 수행 시간
설명	Shell Command는 빠르게 처리될수록 좋다.	
환경	Shell Mode를 수행하는 상태	
자극	User 는 수행을 원하는 Shell Command 를 입력한다.	
반응	Boot Loader 는 Shell Command 를 수행하고, User 에게 결과를 리턴한다.	
측정	[Shell Command 수행 시간] = [Shell Command 입력 시각] – [Shell Command 결과 리턴 시각]	

2.3.4. QA_04 커널 이미지 로딩 가용성

QA_04	Availability	커널 이미지 로딩 가용성
설명	Boot Loader 수행 과정에서 장애 발생 시, 정상 상태로 빨리 돌아갈 수 있어야 한다.	
환경	Boot Loader가 커널 이미지 로딩 진행 중인 상태	
자극	Boot Loader 는 수행 중 장애 상황을 발견한다.	

반응	1. Boot Loader 는 커널 이미지 로딩을 수행한다. 2. Boot Loader 는 커널 이미지 로딩 과정에서 장애 상황을 발견한다. 3. Boot Loader 는 장애 복구를 시도한다. 4. Boot Loader 는 사용자에게 "부팅 과정에서 xx 장애가 발생하였지만, 정상적으로 복구하여 부팅을 진행합니다."와 같은 적절한 메시지를 출력한 후, 부팅을 재개한다.
측정	[정상 복구 시간] = [Recover 완료 시각 (반응 4)] – [Fail 발생 시각 (반응 2))]

2.3.5. QA_05 Shell Command 추가 및 확장 용이성

QA_05	**Modifiability**	**Shell Command 추가 및 확장 용이성**
설명	Shell Command가 추가되거나 확장되었을 때, 시스템의 변경을 위한 개발 비용이 최소화되어야 한다.	
환경	개발 완료 이후 시점 혹은 해당 요구사항 구현 진행 중인 시점	
자극	Shell Command 에 대한 요구사항 변경 혹은 신규 요구사항 요청	
반응	요구사항을 만족하도록 Boot Loader 를 수정 개발한다.	
측정	[개발 비용 (M/M)] = [수정, 검증을 다시 해야 하는 모듈] / [파일의 크기(LoC)]	

3. 시스템 구조

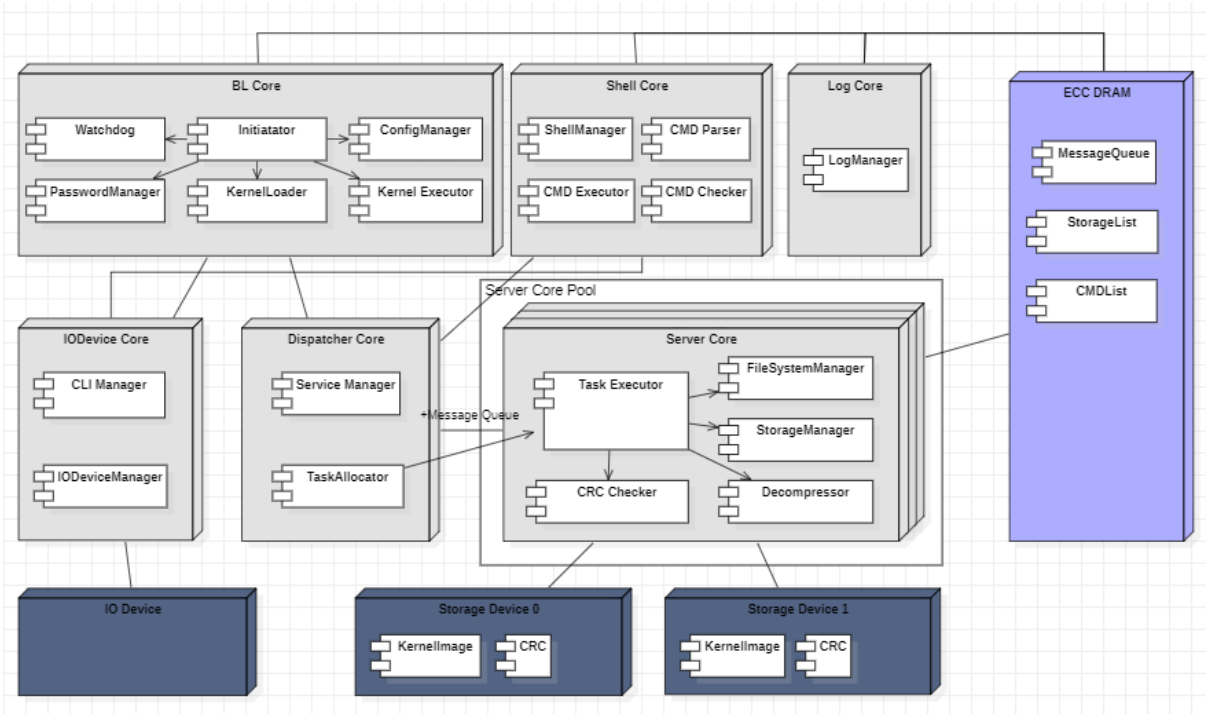

그림 7. 동작 측면에서의 Deployment View.

본 문서에서 제안하는 Boot Loader의 동작 실행 측면의 구조는 위 그림과 같다. 5개의 Dedicated Core와 Server Core Pool에 속하는 Core들로 구성된다. 각 Core의 역할을 다음과 같다.

1. BL Core: Boot Loader 핵심 서비스 로직들을 처리하는 Core이자 Dispatcher Core에게 서비스를 요청하는 Client Core.
2. Shell Core: Shell Mode 지원 및 사용자의 Shell Command 입력에 대한 처리를 위한 Core
3. Log Core: Log Message 출력을 위한 전담 Core
4. IO Device Core: IO Device 관리 전담을 위한 Core
5. Dispatcher Core: Server Core에서 수행할 서비스들을 관리하고, Client Core와 Server Core 간의 채널을 연결해주는 Core. Client Core 부하 감소를 위한 목적.
6. Server Core Pool: 동일 Task를 병렬화 수행하는 Core들의 집합. Server Core들 간의 Task 균등 분배를 위해, Server Core Pool 형태로 운영.

3.1. UC_01에 대한 동작 상세

아래 그림은 UC_01에 대한 시스템 관점에서의 Sequence Diagram이다. 본 장에서는 이를 바탕으로 각 단계에 대한 Core별 상세 동작 구조를 기술한다.

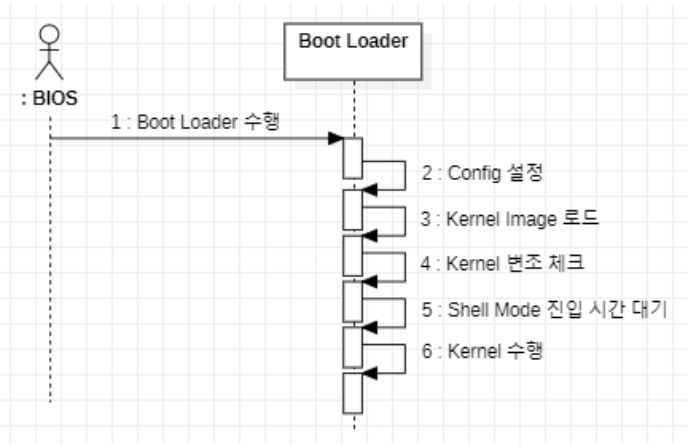

그림 8. 시스템 관점의 Kernel Load Sequence Diagram.

3.1.1. Config 설정

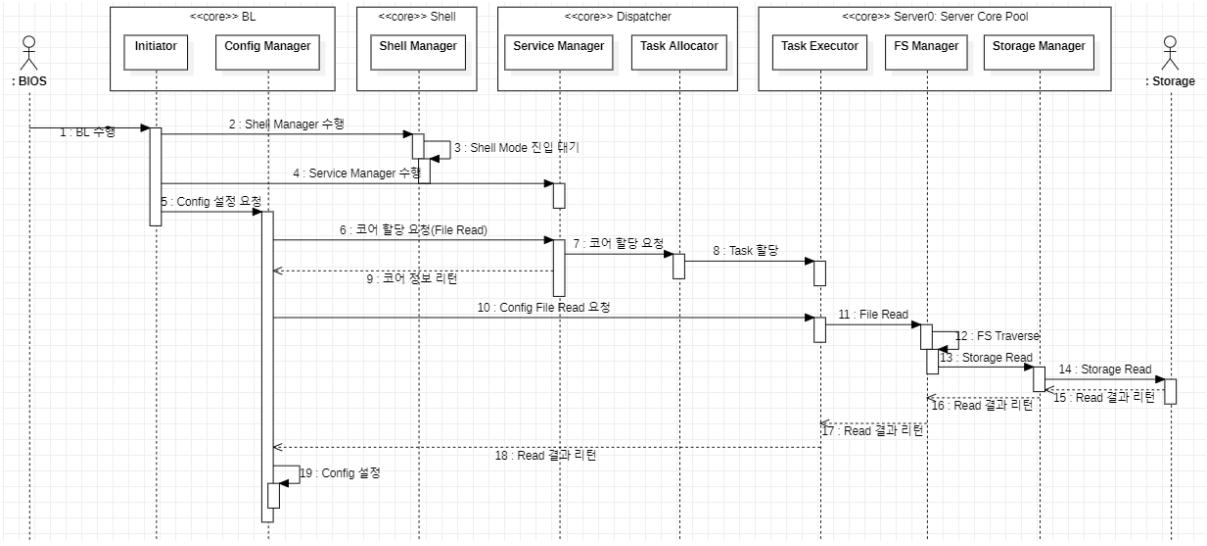

그림 9. Config 설정 Sequence Diagram.

위 그림은 Boot Loader가 수행되었을 때부터 Config 설정 완료하기까지의 Sequence Diagram을 나타낸다. Boot Loader가 수행되면, Initiator는 가장 먼저 Shell Mode 진입 시점을 앞당기기 위해,

Shell Manager를 수행시킨다. Shell Core는 Shell Manager 수행을 완료한 후, Shell Mode 진입 대기를 위한 타이머를 시작한다. Initiator는 Config Manager를 통해, Config 설정을 요청한다. Config Manager는 File System으로부터 Config File을 읽어 오기 위해, Dispatcher Core의 Service Manager에게 File Read 코어 할당을 요청한다. Dispatcher Core는 Server Core Pool 중 하나의 Server Core에게 관련 Task를 할당하고, 해당 정보를 Config Manager에게 알려준다. Config Manager는 할당된 Server Core에게 Task 수행을 요청하며, 결과를 넘겨 받아, Config 설정을 완료한다.

3.1.2. Kernel Image 로드

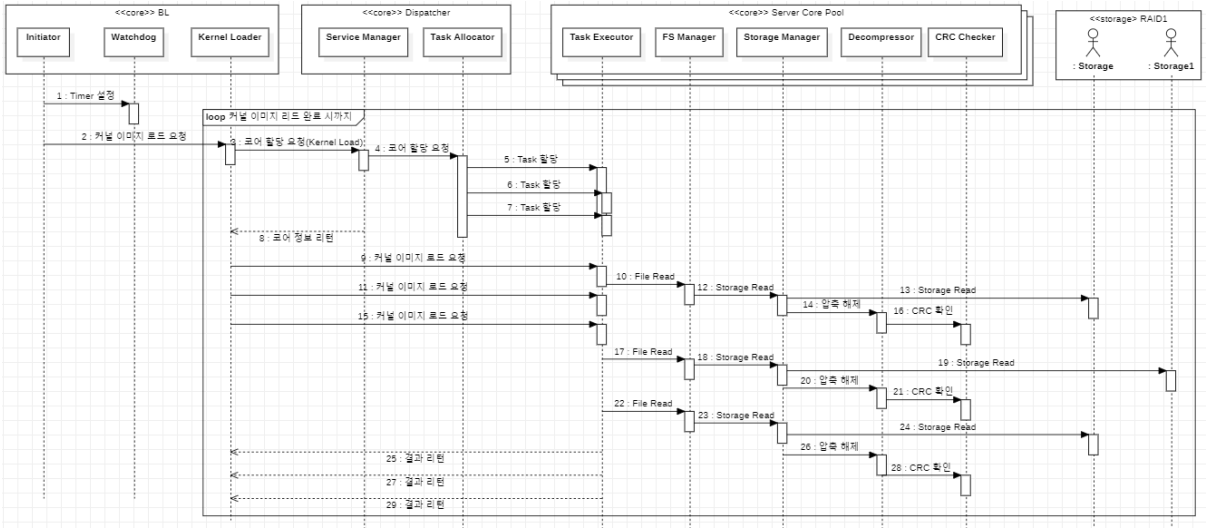

그림 10. Kernel Image 로드 Sequence Diagram.

위 그림은 Initiator가 커널 Image 로드 수행을 위한 Sequence Diagram을 나타낸다. Kernel Image 로드는 Multi-Core를 사용한 병렬화로 성능을 _향_상_시킨다. 따라서 예상치 못한 Hang을 대비하기 위해, Initiator는 Watchdog에게 Timer 설정을 요청한다. 그후, Kernel Loader에게 커널 이미지 요청을 하며, Kernel Loader는 Dispatcher Core의 Service Manager에게 코어를 할당 받는다. 이 때, Task Allocator는 Kernel Loader가 요청한 Task에 대해 Unit Operation (Ex. 4KB Read)으로 나누고, 수행에 필요한 수만큼의 Server Core를 Pool에서 할당해준다.

Kernel Loader는 할당 받은 Server Core들에게 각각 커널 이미지 로드를 요청하며, 각 Server Core들은 압축된 커널 이미지를 읽어와, 압축 해제 및 CRC 체크 후, 그 결과를 리턴 해준다. 이 때, RAID1 구성되어 있는 2개의 Storage로부터 Server Core들이 병렬적으로 커널 이미지를 읽어온다.

3.1.3. Kernel 변조 체크

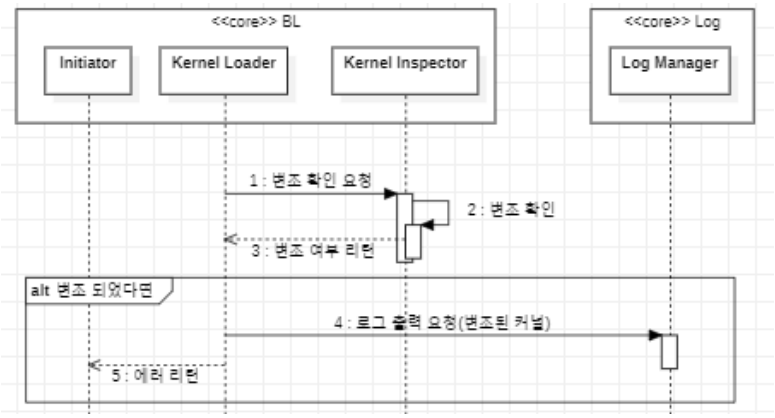

그림 11. Kernel 변조 체크 Sequence Diagram.

위 그림은 Kernel 변조 체크에 대한 Sequence Diagram이다. Kernel Image 로드가 완료되면, Kernel Loader는 Kernel Inspector에게 커널 이미지 변조 확인을 요청한다. Kernel Inspector는 변조 여부를 확인하며, 그 결과를 리턴 한다. 만약, 커널 이미지가 변조되었다면, Kernel Loader는 Log Core의 Log Manager에게 로그 출력을 요청하며, Initiator에게 에러를 리턴 한다.

3.1.4. Shell Mode 진입 대기

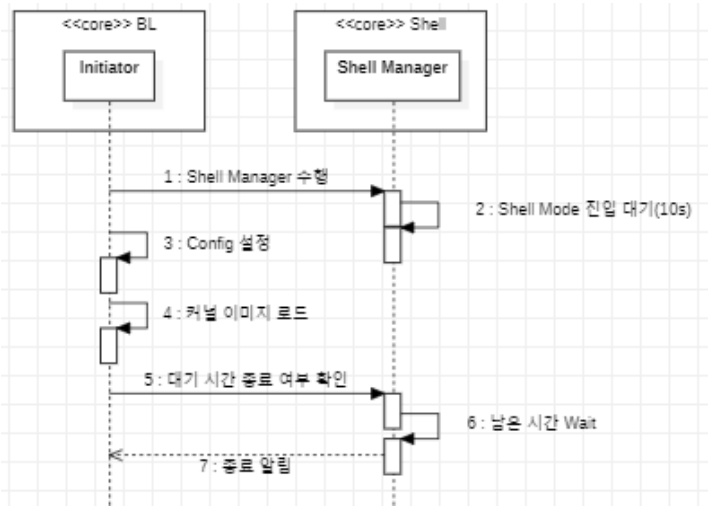

그림 12. Shell Mode 진입 대기 Sequence Diagram.

위 그림은 Shell Mode 진입을 위한 대기 시간을 확인하기 위한 Sequence Diagram.이다. Initiator는 Kernel 수행 전, 사용자가 Shell Mode 진입키를 누르기 위해 특정 시간 (10s) 대기하여야 한다. 따라서 해당 시간을 앞당기기 위해, Initiator는 가장 먼저 Shell Manager를 수행한다. Initiator는 모든 부팅 준비가 끝난 시점에 Shell Manager에게 대기 시간이 완료되었는지를 체크한다. 만약 대기 시간이 남아 있다면, Shell Manager는 해당 시간을 Wait한 후, 종료 후, Initiator에게 알려준다.

3.1.5. Kernel 수행

Initiator는 부팅에 필요한 모든 동작을 마무리하고, Kernel에게 제어권을 넘긴다. 해당 동작은 Initiator가 수행 중인 BL Core에서 진행된다.

3.2. UC_02에 대한 동작 상세

아래 그림은 UC_02에 대한 시스템 관점에서의 Sequence Diagram이다. 본 장에서는 이를 바탕으로 각 단계에 대한 Core별 상세 동작 구조를 기술한다.

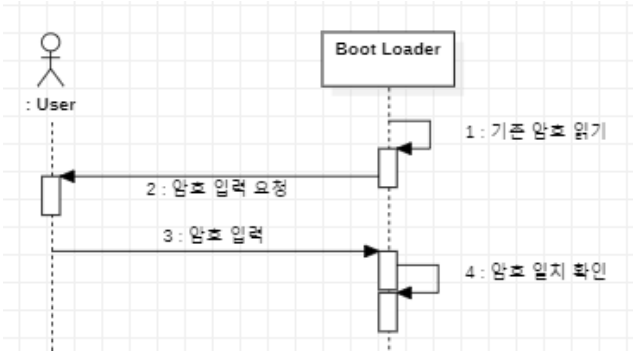

그림 13. 시스템 관점의 Password 입력 Sequence Diagram.

3.2.1. 기존 암호 읽기

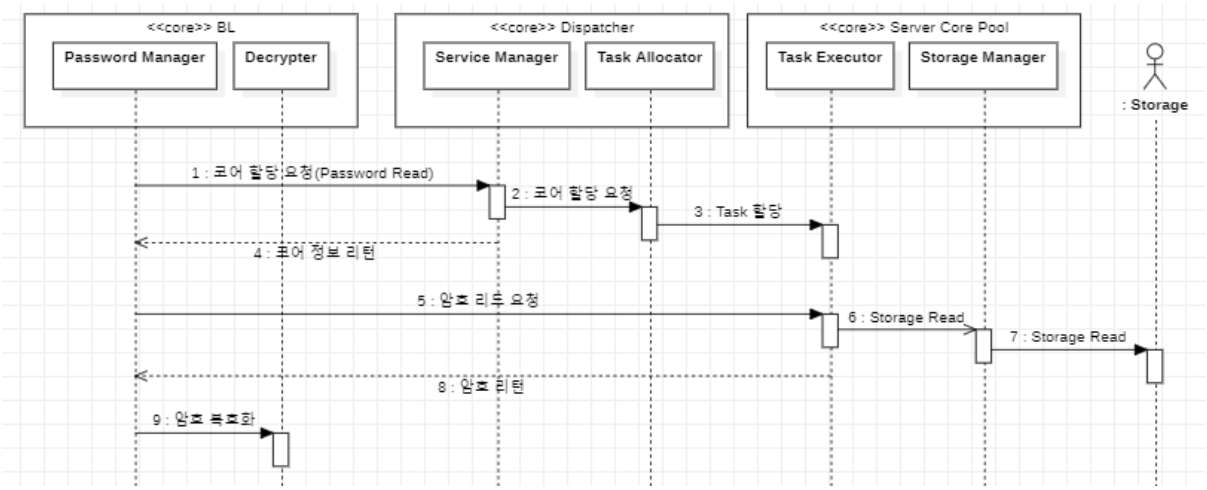

그림 14. 기존 암호 읽기 Sequence Diagram.

위 그림은 Password Manager가 Storage 장치에 저장된 기존 암호를 읽어오는 Sequence Diagram 이다. Password Manager는 Storage 접근을 위해 Dispatcher Core에게 코어 할당을 요청한다. 이후, 할당 받은 Server Core에 암호 리드를 요청하며, 암호 리드가 끝나면, 이를 복호화한다.

3.2.2. 사용자 암호 입력 및 일치 확인

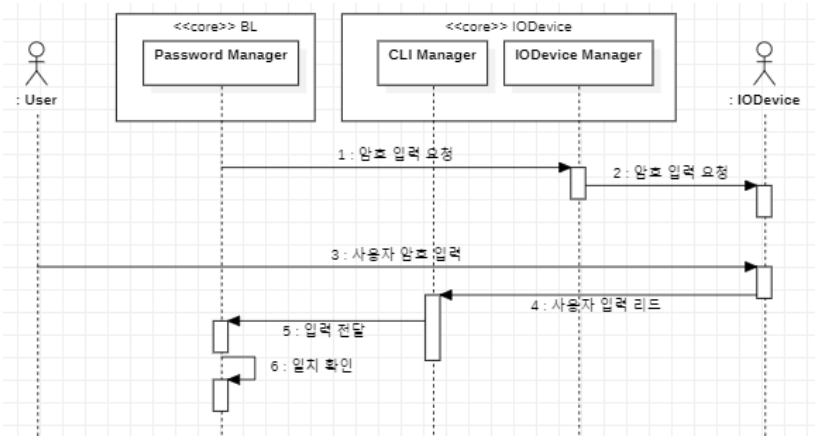

그림 15. 사용자 암호 입력 Sequence Diagram.

위 그림은 Password Manager가 사용자에게 암호 입력을 요청하고, 사용자가 입력한 암호를 IODevice Manager로부터 받아, 일치 여부를 확인하는 Sequence Diagram이다. IODevice Core는

모니터를 통해, 사용자에게 암호 입력 요청을 하고, 키보드를 통해 사용자가 입력한 암호를 리드한다.

3.3. UC_03에 대한 동작 상세

아래 그림은 UC_03에 대한 시스템 관점에서의 Sequence Diagram이다. 본 장에서는 이를 바탕으로 각 단계에 대한 Core별 상세 동작 구조를 기술한다.

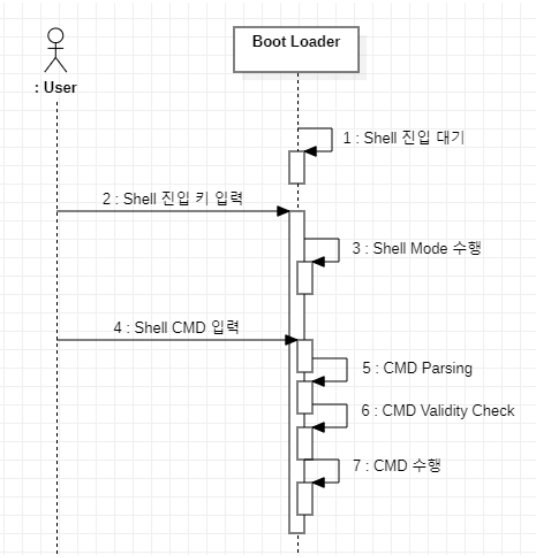

그림 16. 시스템 관점의 Shell Command 수행 Sequence Diagram.

3.3.1. Shell Mode 진입 및 커맨드 입력

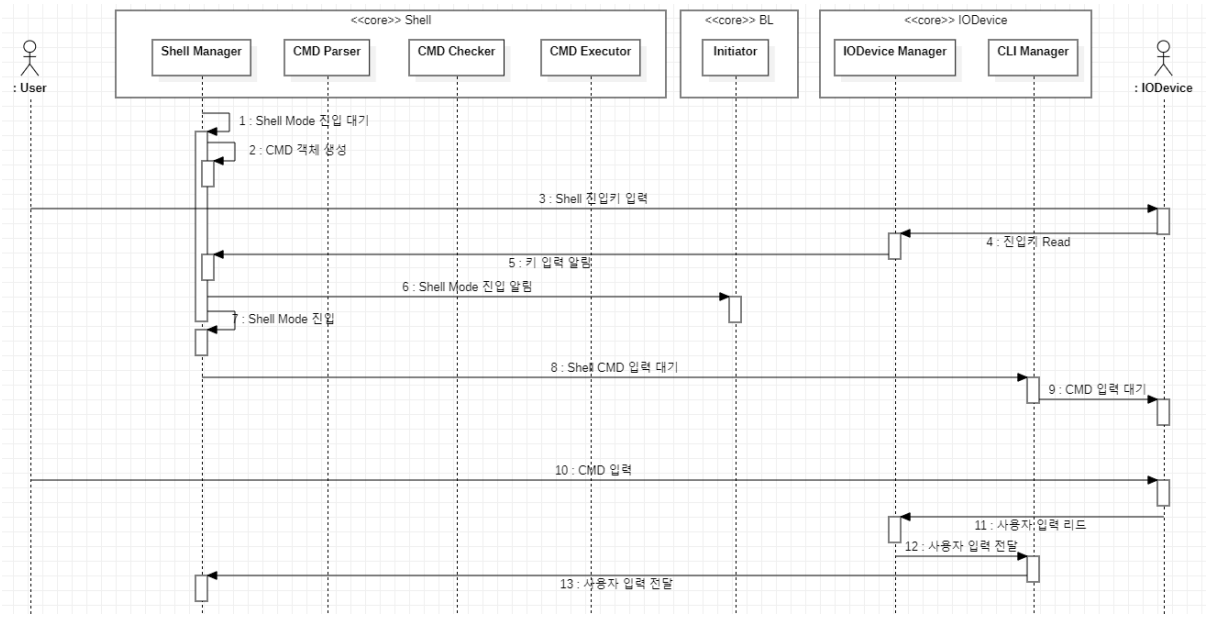

그림 17. Shell Mode 수행 Sequence Diagram

위 그림은 Shell Manager가 Shell Mode 진입 대기 과정에서 사용자가 진입키를 입력하여, Shell Mode로 진입하는 Sequence Diagram이다. Shell Manager는 Boot Loader 시작 시점에 수행되며, Shell Mode 진입을 대기 중이며, 이 과정에서 사용자가 키보드를 통해 Shell 진입키를 입력하면, 이를 IODevice Core로부터 전달받아, Shell Mode로 진입한다. 그 후, 사용자가 입력한 Shell Command는 IODevice Manager, CLI Manager를 거쳐 Shell Manager로 전달된다.

3.3.2. Shell Command 수행

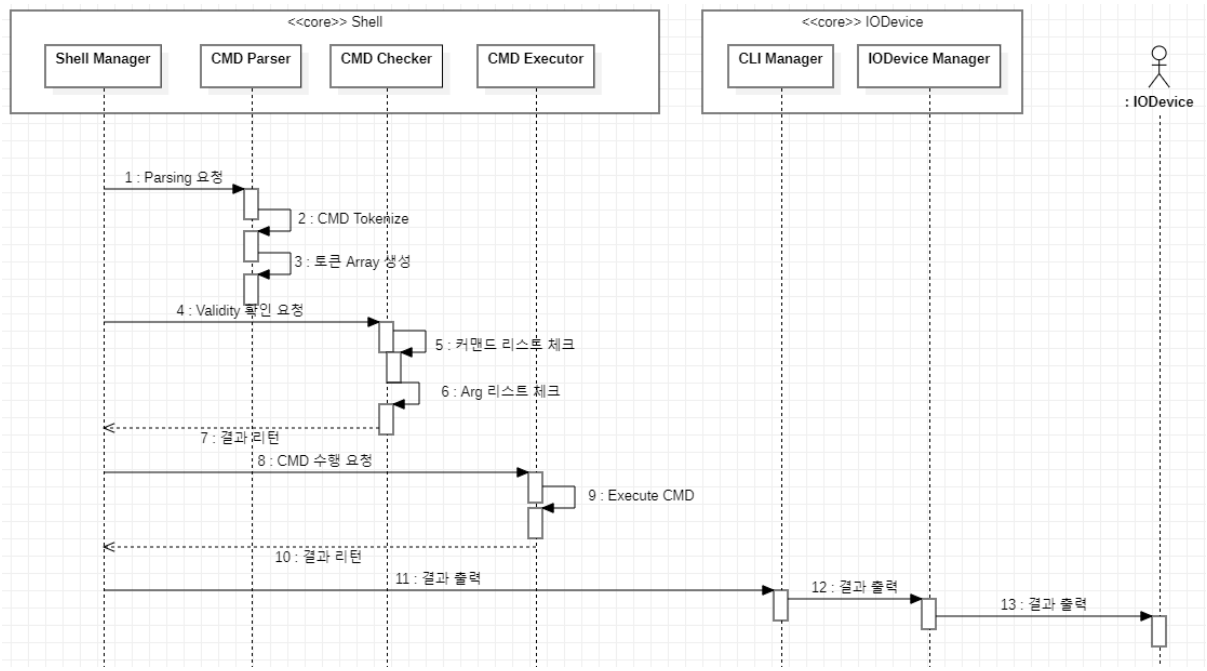

그림 18. Shell Command 수행 Sequence Diagram

위 그림은 Shell Manager가 입력 받은 사용자 커맨드를 수행하는 Sequence Diagram이다. Shell Manager는 CMD Parser에게 Tokenize및 토큰 Array 생성을 요청하며, 그 결과를 바탕으로 CMD Checker에게 유효성 확인을 요청한다. 그 후, CMD Executor를 통해, 커맨드를 수행하고, IODevice를 통해 결과를 제공한다.

4. 모듈 사양

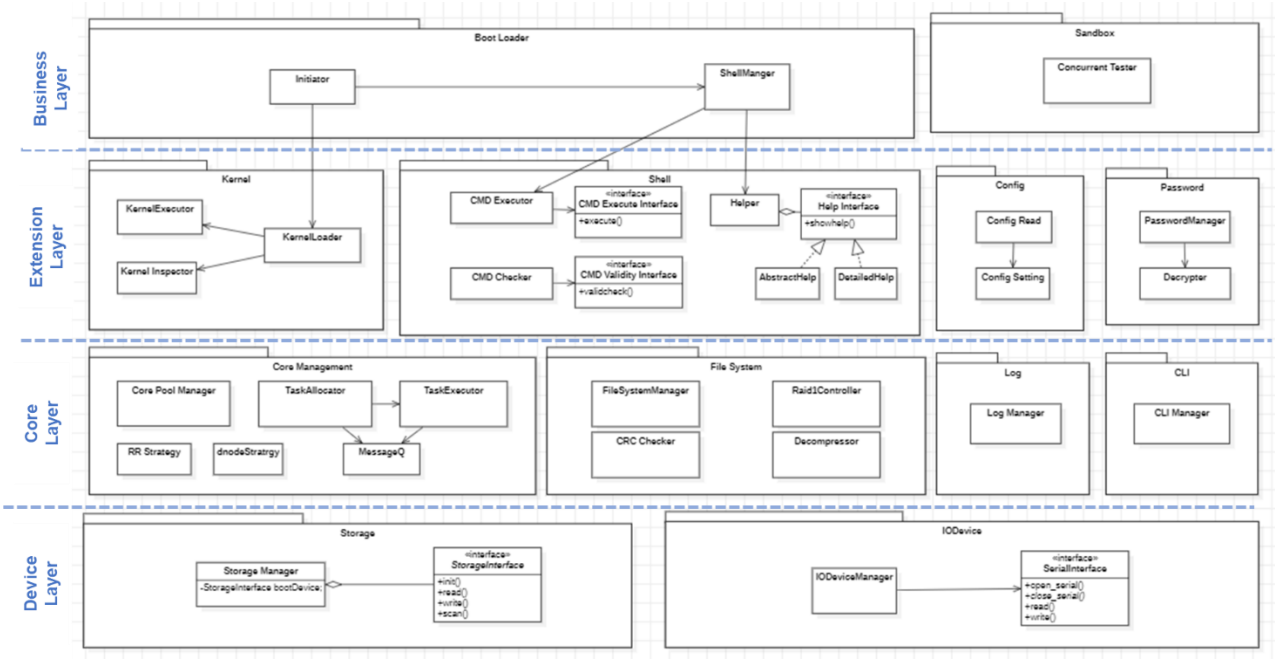

그림 14. 개발 측면에서의 Module View.

위 그림은 본 과제에서 제안하는 Boot Loader의 Module View이다. Layered Style로 구성되어 있으며, 4개의 Layer가 있다. 각 Layer에 대한 설명은 다음과 같다.

- Device Layer: 가장 하단 위치의 Layer로 경계인 Storage와 IO Device 접근에 필요한 인터페이스를 제공한다.

- Core Layer: Device Layer에 속하는 모듈들을 직접적으로 참조하는 모듈들이 속하는 Layer로, 상위 Layer가 필요한 병렬 처리, 파일 시스템 접근, 로그 출력, 사용자 커맨드 처리 관련 기능을 제공하는 모듈들이 위치한다.

- Extension Layer: Core Layer 모듈들을 제공하는 기능과 인터페이스를 통해, 확장된 기능을 제공하는 Layer로, Kernel 로딩 및 Kernel 수행을 담당하거나, Shell Command에 대한 부문 동작들을 수행하는 모듈들이 위치한다.

- Business Layer: 전체 Business Logic을 제공하는 최상위 Layer

4.1. Device Layer

Device Layer는 Storage와 IO Device Package로 구성된다.

4.1.1. Storage Package

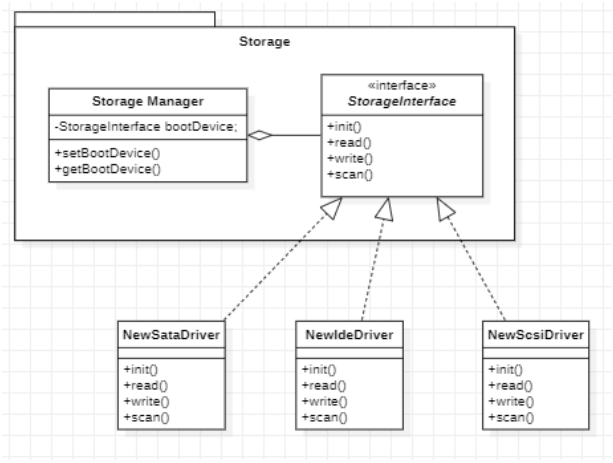

그림 20. Storage Package.

Storage 장치 관리 및 접근을 위한 모듈들이 모여 있다. Storage Interface는 Storage 장치 추가 및 변경이 용이하기 위해, De-factor Standard 인터페이스를 채택하였다. 제조사가 제공하는 스토리지 Driver는 이를 상속받아 구현되어 있다. 추가로, 사용자가 Shell Command를 통해 다른 Storage 장치 선택을 지원하기 위해, 서버에 장착된 Storage 장치에 대한 객체들을 관리한다. 해당 객체들은 Storage Device Scan 시점에 생성되며, 빠른 탐색을 위해 Hash Table로 관리된다.

4.1.2. IODevice Package

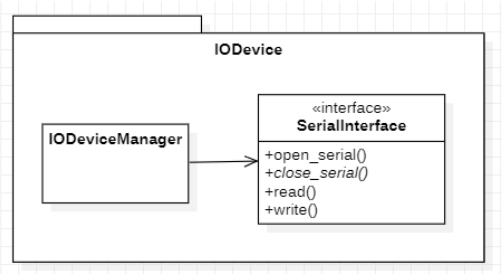

그림 21. IODevice Package.

키보드, 모니터 등 직렬 포트를 통한 사용자 입출력 장치를 위한 모듈들이 모여 있다. Serial Interface는 IODevice 추가 및 변경이 용이하기 위해, De-factor Standard 인터페이스를 채택하였으며, 제조사가 제공하는 IODevice Driver는 이를 상속받아 구현되어 있다.

4.2. Core Layer

Core Layer는 Core Management, File System, Log, CLI Package로 구성된다.

4.2.1. Core Management Package

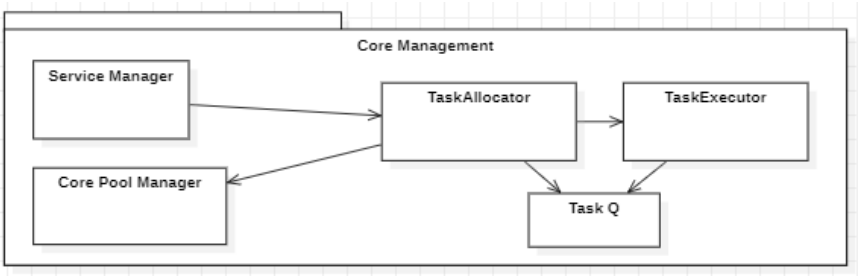

그림 22. Core Management Package.

Multi-Core를 활용한 병렬처리 기능을 위한 모듈들이 위치한다.

- Core Pool Manager는 다수의 Server Core들은 Pool을 관리한다. Dedicated Core를 제외한 나머지 모든 Core에 대한 정보를 가지고 있다.
- Service Manager는 Client로부터 요청이 오면, Task Allocator에게 실제 Task 할당을 의뢰하고, 할당된 Core 정보를 Client에게 전송한다.
- Task Allocator는 다수의 Server Core에게 Task를 배정하는 역할을 수행한다. 이 때, Server Core 간의 균등한 Task 배분을 위해, 기 정의된 Unit Operation (Filesystem Traverse: dnode 기준, Storage Read: 4KB)으로 Task를 나눈다. Server Core에게 Task Queue를 통해, Task를 배정하며, Round Robin 순서로 배정한다.

4.2.2. File System Package

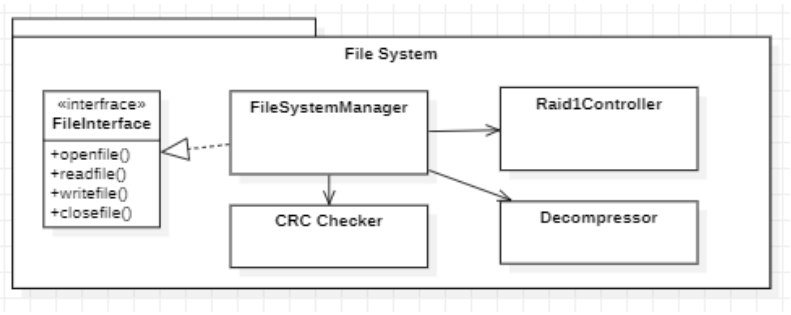

그림 23. File System Package.

상위 Layer에게 File Interface를 통해, File 접근 기능을 제공하는 모듈들이 위치한다. File Read 시, RAID1으로 구성된 2개의 스토리지에서 병렬적으로 데이터를 읽어온다. 추가로, 압축된 커널 이미지의 경우, 압축 해제를 수행하며, CRC 확인 결과 이상이 없는 파일 리드 결과만 전달한다.

4.2.3. Log Package

IODevice를 통해, Log 출력 기능을 담당하는 패키지이다. Log는 각 모듈별로 이미 정의되어 있으며, Log Manager는 Log ID를 통해, Log 출력을 요청 받는다.

4.2.4. CLI Package

IODevice를 통해 키보드로부터 사용자 입력을 받아, 이를 문자열 형태로 전달하는 동작이 구현되는 패키지이다.

4.3. Extension Layer

Extension Layer는 Kernel과 Shell, Config, Password Package로 구성된다.

4.3.1. Kernel Package

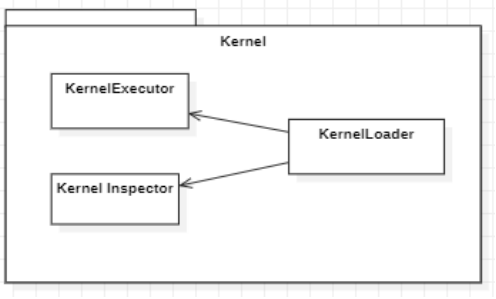

그림 24. Kernel Package.

커널 로딩을 담당하는 모듈들이 모여 있는 패키지이다. Kernel Loader는 커널 로딩 및 커널에게 제어권을 넘겨주기 위한 로직을 담당하는 모듈이다. Config Manager가 설정한 위치의 커널 이미지를 DRAM으로 읽어 온 후, Kernel Inspector를 통해 Kernel 변조 여부 확인하고, 이상이 없을 경우, Kernel을 수행한다.

4.3.2. Shell Package

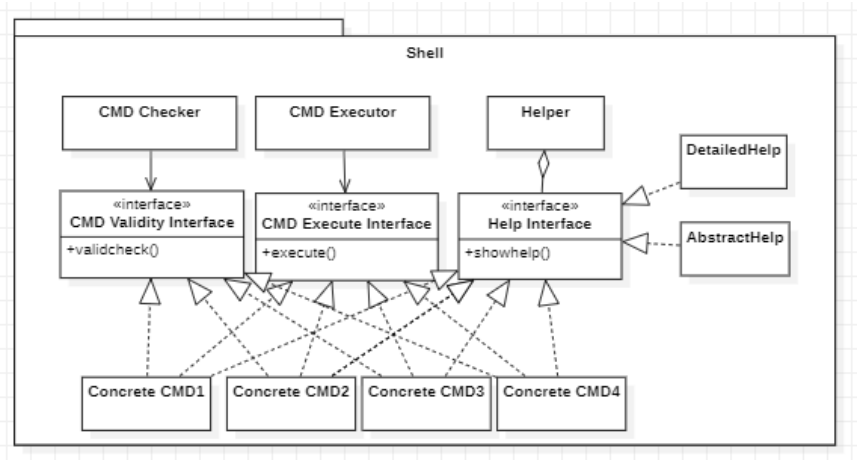

그림 25. Shell Package.

Shell Command 수행을 위해 필요한 부문 동작들을 구현된 패키지이다. 커맨드 수행을 위해 공통으로 필요한 인터페이스는 총 3가지로 각각에 대해 ISP를 적용하여 각각의 모듈로 분리하였다. Concrete Command Class는 총 3가지의 인터페이스를 다중 상속받아 구현하여야 한다. Helper 모듈의 경우, 사용자의 입력에 따라 간략 도움말과 상세 도움말 제공을 위해 전략 패턴을 적용하였

다.

4.3.3. Config Package

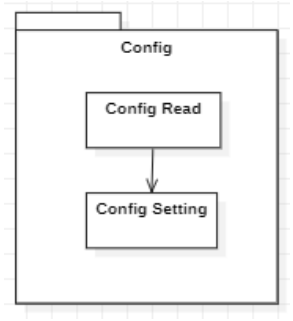

그림 26. Config Package.

Config 설정을 위한 동작들이 구현된 패키지이다. 변경이 빈번한 Config 항목에 대한 부분을 Config Setting 모듈로 분리하였다. Config Read 모듈은 Config 파일로부터 값을 읽어와 Config Setting에 전달한다.

4.3.4. Password Package

사용자 암호 입력에 관련된 동작들이 구현된 패키지이다. Boot Loader 수행 과정에서 사용자 암호가 설정되어 있으면, 사용자로부터 암호를 입력 받아, 해당 암호가 일치하는지 여부를 확인한다. 이 때, 기존 암호는 Encryption된 상태로 저장되어 있으며, 따라서 스토리지 장치로부터 읽어온 암호는 Decryption하여야 한다.

4.4. Business Layer

Business Layer는 Boot Loader Package가 존재한다.

4.4.1. Boot Loader Package

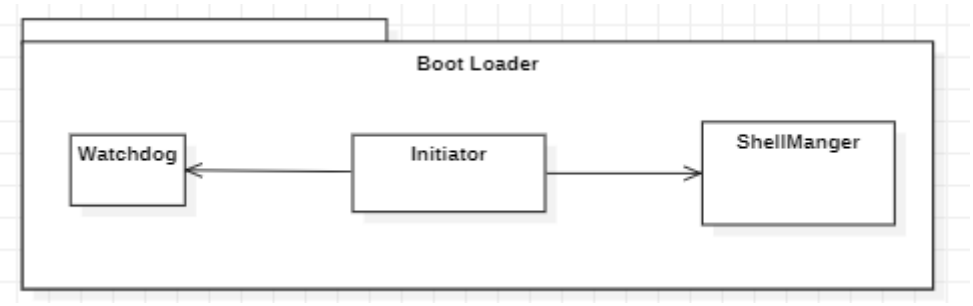

그림 27. Boot Loader Package.

- Initiator: Boot Loader 서비스 위해 필요한 Sequence를 명세하고, Extension Layer에 속하는 모듈을 호출하는 모듈. 예상치 못한 Hang 발생에 대한 빠른 Detect을 위해, 병렬 Core 처리 전 Watchdog Timer를 세팅한다.
- Shell Manager: Shell Mode 제공을 위해 필요한 Sequence를 명세하고 있는 모듈.