Journal of KIIT. Vol. 23, No. 1, pp. 231-240, Jan. 31, 2025. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2025.23.1.231

NFT 기반 티켓 양도 시스템에서의 자바스크립트 런타임 환경 성능 비교 평가

박지안*, 정동원**[1], 정현준**[2]

Comparative Evaluation of Performance of JavaScript Runtime Environments in NFT-based Tickets Transfer Systems

Jian Park*, Dongwon Jeong**[1], and Hyunjun Jung**[2]

이 연구는 정부(과학기술정보통신부)의 재원으로 한국 연구재단의 지원을 받아 수행되고 있습니다
(No. NRF-2022R1G1A1008493)

요 약

이 논문에서는 NFT 기반 티켓 양도 시스템의 안정성을 향상시키기 위해 Node.js와 Bun 두 자바스크립트 런타임 환경의 성능을 트랜잭션 처리와 데이터베이스 질의 처리로 나누어 비교 분석하였다. 실험 결과, 단일 트랜잭션 처리 성능에서는 Bun이 Node.js에 비해 2.41배 높은 성능을 보였고 부하 테스트의 경우 모든 부분에서 Bun이 Node.js보다 우수한 결과를 보였다. 데이터베이스 질의 처리 성능에서는 Node.js가 Bun에 비해 1.19배 차이로 우수한 결과를 나타낸다. 트랜잭션 처리 속도와 고부하 환경에서의 안정성을 고려한다면 NFT 기반 티켓 양도 시스템을 설계할 때 Bun을 사용하는 것이 적합하다고 판단되지만, 데이터베이스 질의 처리 성능은 한계점으로 존재하며, 이후 Bun의 업데이트를 통해 최적화된 라이브러리나 드라이버를 도입한다면 충분히 보완 가능하다. 결과적으로, 이 논문에서는 NFT 기반 티켓 양도 시스템의 성능 개선을 위해서는 Node.js보다 Bun의 활용이 적절함으로 보였다.

Abstract

In this paper, in order to improve the stability of the NFT-based ticket transfer system, the performance of the two JavaScript runtime environments, Node.js and Bun, was compared and analyzed by dividing it into transaction processing and database query processing. As a result of the experiment, Bun showed 2.41 times higher performance than Node.js in single transaction processing performance, and Bun showed better results than Node.js in all parts of the load test. In database query processing performance, Node.js shows superior results by 1.19 times compared to Bun. Considering the transaction processing speed and stability in a high-load environment, it is considered appropriate to use Bun when designing an NFT-based ticket transfer system, but the database query processing performance exists as a limitation, and it can be sufficiently supplemented if an optimized library or driver is introduced through Bun's update. As a result, in this paper, it seemed that the use of Bun was more appropriate than Node.js to improve the performance of the NFT-based ticket transfer system.

Keywords

JavaScript runtime environments, Node.js, bun, NFT-based ticket system

* 국립군산대학교 소프트웨어학과
- ORCID: https://orcid.org/0009-0009-9404-1770
** 국립국산대학교 소프트웨어학과 교수(교신저자)
- ORCID[1]: https://orcid.org/0000-0001-9887-5336
- ORCID[2]: https://orcid.org/0000-0002-6717-1395

· Received: Oct. 27, 2024, Revised: Nov. 21, 2024, Accepted: Nov. 23, 2024
· Corresponding Author: Dongwon Jeong, Hyunjun Jung
Dept. Software at Kunsan National University, 558, Daehak-ro, Kunsan-si, Jeollabuk-do, Republic of Korea
Tel.: +82-63-469-8917, Email: {junghj85, djeong}@kunsan.ac.kr

Ⅰ. 서 론

현재 아이돌 콘서트, 뮤지컬 등 많은 분야에서 온라인 티켓 암표 거래가 활성화되고 있다. 이러한 티켓 재판매가 문제가 되는 이유는 위조한 공연 티켓 등으로 사기 당하거나 원래 정가보다 높여 판매된 티켓을 통해 발생하는 이익을 본 판매자가 아닌 티켓 재판매자가 갖는 데 있다[1].

온라인에서 티켓 재판매를 처벌할 수 없는 이유는 다음과 같다. 관련 조항이 있는 경범죄 처벌법, 공연법을 살펴보자. 경범죄 처벌법 제3조에 포함된 항목에는 '암표 매매'가 포함된다. 하지만 해당 조항의 전문을 살펴봤을 때, '현장거래'에 한정된 조항이기 때문에 온라인 거래는 포함이 되지 않는다. 또한 공연법에는 '암표'라는 단어 대신 제4조의2 제1항에 '부정판매'라는 단어가 나올 뿐, 부정 판매에 대한 처벌 조항 자체가 없다. 따라서 온라인에서는 어떤 식으로 티켓을 판매해도 처벌할 수 있는 확실한 방법은 없다[2].

티켓 예매처의 경우에는 자체적으로 매크로 적발 수단을 도입하고 있지만 완벽하지 않아 티켓 재판매 업자들이 충분히 피해갈 수 있는 구조이다. 오프라인의 경우에는 한국에서 진행하는 공연은 '본인 명의로 된 ID로 티켓 예매를 하고 본인만 콘서트에 참여할 수 있다'라는 원칙을 기본으로 한다. 이에 현장에서는 신분증, 사용자 ID, 티켓 예매창 등을 확인하여 입장을 허용하며 하나라도 증명되지 않을 경우 입장을 불허한다[3].

티켓 예매처와 오프라인에서 실행하는 암표 단속 수단은 결국 관객들에 대한 통제만 강화할 뿐 근본적인 해결책은 되지 못한다. 이에 NFT 기반 티켓을 통해 티켓 발행 및 소유권 변경 내역을 투명하게 기록하여 거래의 신뢰성을 향상시키는 시스템이 제안되었다[4]. 시스템은 티켓 발급과 재판매 과정에서 사용자 요청이 동시에 발생하며, 블록체인 트랜잭션 처리, 데이터베이스 질의, 실시간 인증 등 복잡한 연산이 요구되며 서버 성능은 핵심적인 요소로 작용한다. 때문에 서버 성능이 떨어지면 처리 지연이나 시스템 불안정이 발생하여 시스템의 신뢰성을 떨어트린다. 기존 제안된 시스템에서는 자바스크립트 런타임 환경으로 Node.js를 활용했는데, Node.js는 단일 스레드 방식으로 복잡한 연산을 요구할 때 성능이 저하된다는 단점이 있다.

이 논문에서는 이러한 Node.js의 단점을 극복하기 위해 2023년 9월에 새롭게 출시한 자바스크립트 런타임 환경인 Bun을 적용하고자 한다. Bun을 활용하고자 하는 이유는 다음과 같다. 첫째, Bun은 빠른 실행 속도와 낮은 지연 시간을 제공한다. 둘째, 빌트인 번들러와 패키지 관리자를 포함하여 개발 효율성을 높일 수 있다. 셋째, 최신 기술 도입으로 시스템의 경쟁력을 강화하고 발전을 기대할 수 있다. 따라서 Node.js와 Bun의 성능을 비교하여 더 우수한 런타임을 선택함으로써 효율적이고 안정적인 서버 환경을 구축하고자 한다.

서버 평가는 블록체인 트랜잭션 처리 능력, 데이터베이스 질의 처리 성능을 중점으로 살펴본다. 블록체인 트랜잭션 처리는 두 가지로 나누어 평가한다. 첫 번째는 NFT 티켓 발급 시 트랜잭션 처리를 진행한 속도를 측정하고, 두 번째는 부하 테스트 도구를 활용하여 다수의 사용자가 동시다발적으로 트랜잭션을 발생시키는 상황에서 Bun과 Node.js의 속도를 측정하고 비교한다. 또한 시스템에서는 티켓 정보, 거래 내역 등을 빠르게 조회하고 처리하는 것을 필수로 요구하기 때문에 티켓 조회, 티켓 검색, 사용자 본인 확인 질의를 전송하고 응답 받기까지 시간을 측정하여 비교한다.

Ⅱ. 배경 기술

이 장에서는 Node.js와 Bun에 대해 살펴보고 이 둘의 대표적인 차이점인 자바스크립트 엔진의 동작 과정을 기술한다.

2.1 Node.js

Node.js는 Chrome V8 JavaScript 엔진을 기반으로 한 JavaScript 런타임 환경이다. Node.js의 장점은 V8 엔진의 최적화로 인해 JavaScropt코드를 빠르게 실행하고 프론트엔드와 백엔드 모두에서 사용할 수 있어 개발 생산성이 높다.

하지만 단일 스레드이기 때문에 CPU를 많이 사용하는 작업에서 성능 저하가 발생하고 비동기 코드를 많이 사용할 경우 콜백 함수가 중첩되어 코드 가독성이 저하되는 단점을 지닌다[5][6].

2.2 Bun

Bun는 JavaScript 및 TypeScript를 위한 고성능 런타임 환경으로, 속도와 효율성에 초점을 맞춰 개발됐다. Node.js와 호환성을 유지하면서 더 빠른 실행 속도를 제공한다. Bun는 JavaScriptCore 엔진을 기반으로 하여 빠른 시작 시간과 낮은 메모리 사용량을 제공하고 Zig언어로 작성되어 고성능 네이티브를 구현했다. 또한 내장된 번들러를 통해 별도의 설정 없이도 코드를 번들링 할 수 있어 빌드 속도가 빠르다. 하지만 아직 개발 초기 단계이기 때문에 안정성 검증이 완벽하지 않다는 단점이 있다[7].

2.3 V8엔진과 JSC엔진 비교

그림 1의 A는 V8엔진의 동작 과정을 나타내고, 그림 1의 B는 JSC엔진의 동작 과정을 나타낸다[8].

V8엔진은 구문 분석을 통해 추상 구문 트리(AST)를 생성하고, Ignition 인터프리터로 바이트 코드로 변환한 후, FurboFan JIT 컴파일러가 자주 사용되는 코드를 실시간으로 기계어로 최적화하여 실행하고 가비지 컬랙션을 통해 메모리를 관리한다.

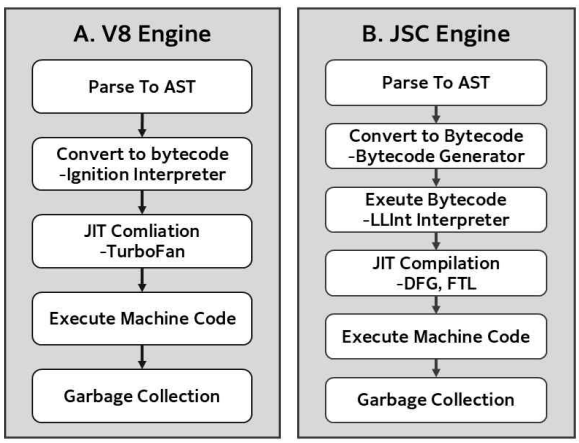

그림 1. V8엔진과 JSC엔진의 동작과정
Fig. 1. Operation processes of V8 engine and JSC engine

JSC(JavaScriptCore)도 유사한 방식으로 구문 분석 후 AST를 생성하고, LLInt 인터프리터를 통해 바이트 코드를 실행한다. JSC는 DFG JIT와 FTL JIT컴파일러를 통해 데이터 흐름을 분석하고 최적화한다. 두 엔진의 차이점은 코드 처리 방식에 있으며, 이는 어플리케이션의 성능 요구에 따라 다르게 영향을 미친다[9][10][11].

III. 시스템 구조

이 장에서는 관련 연구를 살펴보고 비교 실험을 위해 구현한 시스템의 개요를 살펴본다.

3.1 NFT 기반 티켓 양도 시스템 관련 연구

대체 불가능 토큰(NFT, Non-fungible token)란 블록체인 기술을 이용해서 디지털 자산의 소유주를 증명하는 가상의 토큰(Token)으로[12] NFT 기반 티켓은 이를 활용하여 발행한 티켓이다.

[4]에서는 NFT 티켓을 활용하여 티켓의 소유권을 블록체인에 기록하고, 이를 통해 각 티켓의 거래 내역과 소유권을 투명하게 관리할 수 있는 양도 시스템을 개발했다. 스마트 계약을 통해 티켓 구매와 양도 과정에서 자동으로 조건을 실행하여, 양도 시 특정 조건을 만족해야만 거래가 이루어지도록 했다. 또한, 악의적 리셀러 검증 알고리즘을 도입해 실제 구매 이력이 있는 계정만 양도가 가능하게 하여, 암표 거래를 억제하고, 거래의 신뢰성과 보안성을 보장할 수 있게 구성하였다.

[13]에서는 이더리움의 ERC-721 표준을 기반으로 NFT을 통해 티켓의 소유권을 블록체인에 기록하여 각 티켓의 거래 내역과 소유권을 투명하게 관리할 수 있는 시스템을 개발했다.

3.2 Node.js와 Bun의 비교 연구

자바스크립트 런타임 환경이란 프로그램이 실행되는 동안 필요한 모든 요소와 리소스를 포함한 환경으로 주로 메모리 관리, 데이터 관리, 입출력 관리, 라이브러리 지원 등을 담당한다.

[14]에서는 HTTP서버 요청, 독립 실행형 스크립트 실행을 통해 Node.js와 Bun의 성능을 비교한다.

메모리 사용량, 응답 시간, 실행 시간, 요청 처리량 등을 기준으로 두 런타임을 분석했으며, Bun이 전반적으로 더 빠르고 메모리 관리 측면에서 우수하다는 결론이 도출됐다.

또한 [15]에서는 블록체인 웹 애플리케이션 환경에서 Node.js와 Bun의 블록 생성 성능 실험을 통해 비교 및 평가를 진행한다. 실험 결과 Low Level에서는 Bun이 Node.js보다 블록 생성 속도에서 87.59%로 빠른 성능을 보인다는 결론이 도출됐다.

이를 근거로 하여, 이 논문에서 Bun을 사용한 결과가 Node.js를 사용한 결과보다 우수할 것이라고 예측한다.

3.3 전체 시스템 개요

그림 2는 시스템의 개요를 나타낸다. MetaMask 지갑은 사용자가 블록체인 네트워크에 로그인하고 자신의 계정을 인증하며, Web3.js와 연동되어 스마트 계약을 호출하고 트랜잭션에 서명하는 역할을 한다. Bun과 Node.js 서버는 각각 사용자의 요청을 받아 HTTP 요청을 처리한다.

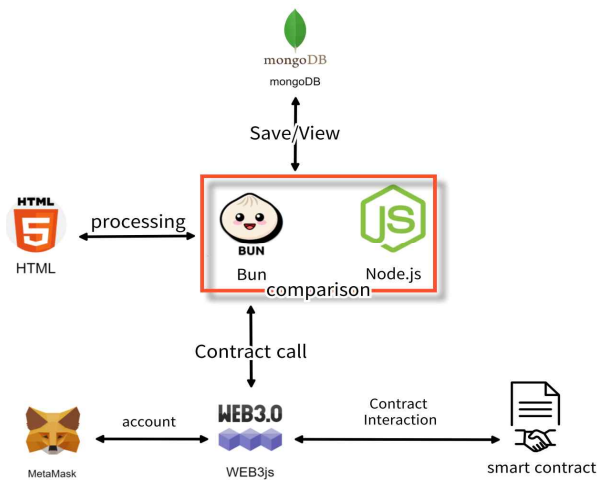

그림 2. 전체 시스템 개요
Fig. 2. Overall system overview

이 논문에서는 MongoDB를 통해 블록체인에는 소유권과 트랜잭션 해시 같은 중요한 정보들만 기록하여 비용을 절감하며, MongoDB에 저장되는 메타데이터는 표 1에 나타난 데이터를 저장한다. Web3.js는 이더리움 블록체인과 상호작용하여 Solidity로 작성된 스마트 계약과 통신하며, NFT 티켓의 발행과 소유권 이전을 관리한다. 생성된 트랜잭션은 Infura API를 통해 Ethereum Sepolia 테스트넷에 전달되어 블록체인에 기록된다. 마지막으로, 표 2는 제안 시스템의 주요 함수를, 표 3은 NFT 티켓을 발급하기 위한 Solidity 코드를 보여준다.

표 1. 제안 시스템의 메타 데이터
Table 1. Metadata for proposed system

Parameter	Data type	Description
price	uint256	Ticket price
purchaseTime	uint256	Ticket purchase time
buyer	address	Ticket buyer's address
buyerDID	string	Buyer's DID
eventName	string	Performance name
eventLocation	string	Performance venue
eventDate	string	Performance date
buyerName	string	Ticket buyer's name
tokenURI	string	URI containing metadata or ticket image
isUsed	bool	Ticket usage status

표 2. 스마트 계약 함수
Table 2. Smart contract functions

Overview	Funtion
Issuer	issuerTicket() getTicketDetails() requestTrade() comleteTrade()
User	viewMyTickets() transferTicket()

표 2는 제안된 시스템의 주요 함수들을 나타낸다. Issuer은 issueTicket() 함수를 사용하여 새로운 티켓을 발행하고 getTicketDetails() 함수를 통해 티켓 상세 정보를 조회할 수 있다. requestTrade() 함수를 사용하여 티켓 거래를 요청하며 completeTrade() 함수를 호출하여 티켓 거래를 완료할 수 있다. User는 viewMyTickets() 함수를 사용하여 자신이 소유한 티켓들을 조회하고 transferTicket()함수를 통해 티켓의 소유권을 다른 사용자에게 이전할 수 있다.

표 3은 스마트 계약 코드 일부를 보여준다. NFT 티켓을 발행하는 함수 issueTicket 함수는 스마트 계약의 소유자가 특정 구매자에게 새로운 NFT 티켓을 발행하고, 해당 티켓의 상세 정보를 저장하며, 발행 사실을 이벤트로 알리는 기능을 수행한다[16].

표 3. 스마트 계약 코드
Table 3. Smart contract code

```
function issueTicket(
    address _buyer,
    string memory _buyerDID,
    uint256 _price,
    string memory _eventName,
    string memory _eventLocation,
    string memory _eventDate,
    string memory _buyerName
)
    public onlyOwner returns (uint256)
{
    uint256 tokenId = totalSupply() + 1;
    _mint(_buyer, tokenId);

    tickets[tokenId] = Ticket(
        _price,
        block.timestamp,
        _buyer,
        _buyerDID,
        _eventName,
        _eventLocation,
        _eventDate,
        _buyerName
    );

    emit TicketIssued(tokenId, _buyer, _price);
    return tokenId;
}
```

IV. 실험 설계

이 장에서는 이 논문에서 자바스크립트 런타임 환경의 성능을 측정하는 실험 과정을 설계한다.

4.1 트랜잭션 수행 실험 설계

본 실험은 Node.js와 Bun에서 실행되며, 과정은 그림 3을 참고한다. 실험 참가자가 NFT 티켓 발급을 위한 트랜잭션을 요청한다. 이 요청은 각 Node.js와 Bun 서버에 전달되며, 서버는 해당 스마트 계약을 블록체인에 호출하여 트랜잭션을 실행한다. 트랜잭션의 처리가 성공적으로 완료되면, 사용자는 시스템으로부터 티켓 발급 완료 메시지를 수신한다. 트랜잭션의 시작부터 완료까지의 전체 소요 시간은 성능 측정을 위해 기록된다.

위와 동일한 상황에서 고부하 상황을 모방하기 위해 다수의 사용자가 동시에 트랜잭션을 요청하는 시나리오를 구성한다. 이 과정에서 최소, 최대, 평균 응답 시간과 세션 길이를 최소, 최대, 평균으로 나누어 기록한다. 마지막으로 기록된 데이터를 바탕으로 분석하여 Node.js와 Bun의 트랜잭션 처리 성능과 시스템 안정성을 비교한다[17].

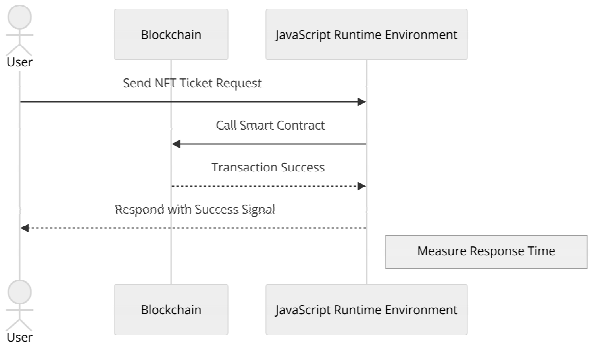

그림 3. 트랜잭션 처리 시퀀스 다이어그램
Fig. 3. Transaction processing sequence diagram

4.2 데이터베이스 질의 처리 실험 설계

이 실험 설계는 NFT 티켓 정보 조회 시, Node.js와 Bun의 데이터베이스 질의 처리 성능을 비교하며, 과정은 그림 4를 참고한다. 사용자는 UI를 통해 티켓 조회, 티켓 검색, 사용자 정보 확인 등 다양한 요청을 할 수 있다. 모든 사용자로부터의 요청은 MongoDB 데이터베이스로 질의를 전송하는 형태로 처리된다. 이 과정은 Node.js와 Bun에서 호환성 및 실행 능력을 평가하는 데이터를 제공한다. MongoDB에서 처리된 질의 결과는 Node.js와 Bun환경으로 각각 반환된다. 반환된 데이터를 기반으로 각 환경에서의 응답 시간이 측정된다.

동일한 조건에서 다수의 질의를 동시에 처리하여 각 환경에서의 최소, 최대, 평균 응답 시간을 기록한다. 수집된 데이터를 통계적으로 분석하여 Node.js와 Bun환경에서의 데이터베이스 질의 처리 성능 차이를 확인할 수 있다.

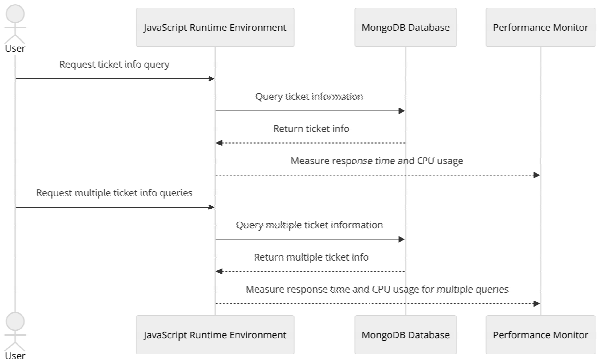

그림 4. 데이터베이스 질의 처리 시퀀스 다이어그램
Fig. 4. Database query processing sequence diagram

V. 구현 및 평가

이 장에서는 구현 환경과 구현된 시스템 화면을 살펴보고 4장에서 설계한 실험을 진행한다. 실험이 완료되면 결과를 정량적 평가와 정성적 평가로 나누어 진행한다.

5.1 구현 환경 및 구현 화면

이 논문에서 제안하는 시스템은 표 4와 같은 환경에서 구현되었으며, Ethereum Sepolia 테스트넷을 활용하여 진행한다.

표 4. 데이터베이스 질의 처리 시퀀스 다이어그램
Table 4. Database query processing sequence diagram

Feature	Specification
OS	Ubuntu 20.04 LTS
CPU	Intel(R) Xeon(R) Gold 6248@ 2.50GHz
Memory	256GB
SSD	2TB

그림 5는 양도 티켓 발행 화면을 나타낸다. 티켓 가격, 이벤트명, 이벤트 날짜 및 위치, 구매자 이름 등이 포함되며, 정보를 입력한 후, 발행 버튼을 누르면 양도 티켓 발행 트랜잭션을 실행한다. 발행된 티켓은 Ethereum Sepolia 테스트넷에 기록되며, 트랜잭션이 성공적으로 처리되면 양도 티켓 발행이 완료된다.

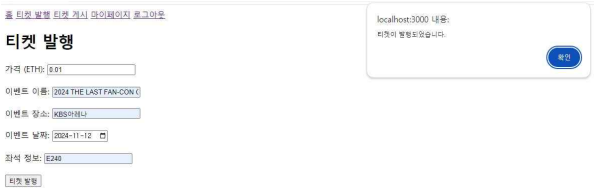

그림 5. 티켓 발행 UI
Fig. 5. UI of ticket issuance

그림 6은 양도 티켓 발행 트랜잭션이 완료된 후 EtherScan을 통해 확인되는 화면이며, 이를 통해 발행된 티켓이 ERC-721 표준에 맞춰 성공적으로 생성된 것을 확인할 수 있다. 사용자는 발행된 티켓을 그림 7처럼 마이페이지를 통해 조회하고 티켓 ID, 이벤트명, 거래 가격을 확인할 수 있다.

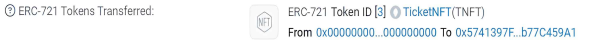

그림 6. 토큰 발행 확인
Fig. 6. Confirmation of token issuance

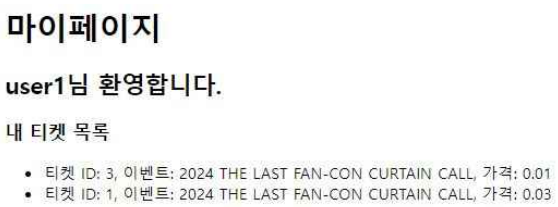

그림 7. 마이페이지
Fig. 7. UI of my page

5.2 정량적 비교 평가: 트랜잭션 처리

이 절에서는 티켓 양도 과정에서 Node.js와 Bun의 트랜잭션 처리 실험을 진행한다. 자세한 실험 설계는 그림 3을 참고한다.

5.2.1 단일 트랜잭션 처리 성능 비교

그림 8의 실험 결과에서, Bun을 사용했을 경우 응답시간은 4.82s를 나타냈고 Node.js를 사용했을 경우 응답시간은 11.619s를 나타냈다.

단일 트랜잭션 처리 성능을 비교했을 때 응답시간에 있어 Bun가 약 2.41배로 상대적으로 빠르다는 것을 확인할 수 있다.

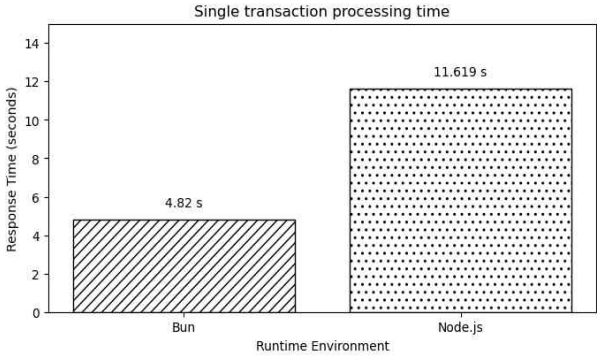

그림 8. 단일 트랜잭션 처리 비교 그래프
Fig. 8. Single transaction processing comparison graph

5.2.2 트랜잭션 부하 테스트 성능 비교

표 5와 표 6는 각각 Bun와 Node.js에서 트랜잭션 부하 테스트를 진행한 결과를 나타낸다. 부하 테스트는 Artillery 도구를 활용하여 진행하였고 60초 동안 매초 100명의 사용자가 진입하는 것을 가정하였으며 총 요청 수는 6000건이다.

표 5. 부하 테스트 Bun
Table 5. Load test Bun

Metric	Value
Minimum response time	1.78s
Maximum response time	23.76s
Average response time	4.02s
Minimum session length	18.1s
Maximum session length	24.01s
Average session length	4.06s

표 6. 부하 테스트 Node.js
Table 6. Load test Node.js

Metric	Value
Minimum response time	5.51s
Maximum response time	99.93s
Average response time	67.82s
Minimum session length	55.5s
Maximum session length	99.96s
Average session length	67.89s

최소, 최대, 평균 응답 시간과 최소, 최대, 평균 세션 길이를 지표로 살펴본 결과 최소 응답 시간은 Bun과 Node.js의 차이가 크게 보이지 않지만 최대 응답 시간, 평균 응답 시간은 Bun이 Node.js에 비해 큰 차이로 빠르다는 것을 보여준다. 또한 최소 세션 길이, 최대 세션 길이, 평균 세션 길이를 측정한 결과 모든 방면에서 Bun이 Node.js보다 우수하다는 것이 보여지며, 모든 방면에서 Bun이 Node.js보다 부하 상황에서의 안정성이 더 높은 것을 확인할 수 있다.

그림 9는 앞서 기술한 비교 평가 결과를 막대 그래프로 보여준다.

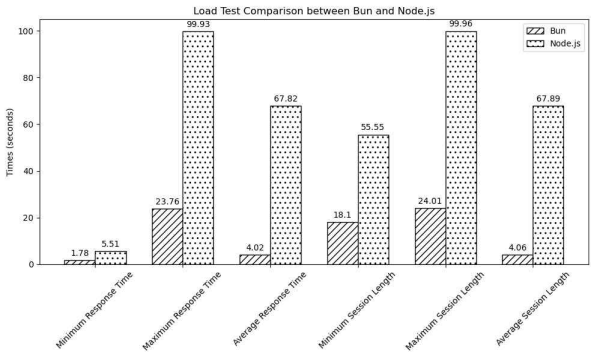

그림 9. 트랜잭션 부하 테스트 비교 그래프
Fig. 9. Transaction load test comparison graph

5.3 정량적 비교 평가: 데이터베이스 질의

그림 10은 데이터베이스 질의 과정에서 Node.js와 Bun의 트랜잭션 처리 시나리오에 따른 실험 결과를 보여준다.

그림 10에서, Bun에서 실험을 진행한 결과 질의에 대한 응답시간은 4.82ms, 평균 응답시간은 17.58ms을 나타냈고, Node.js에서 실험을 진행한 결과 질의에 대한 응답 시간은 6.18ms, 평균 응답시간은 14.77ms이다. 결과를 비교해봤을 때 데이터베이스 질의 처리 성능은 1.19배라는 미세한 차이지만 Node.js가 더 우수한 결과를 나타냈다.

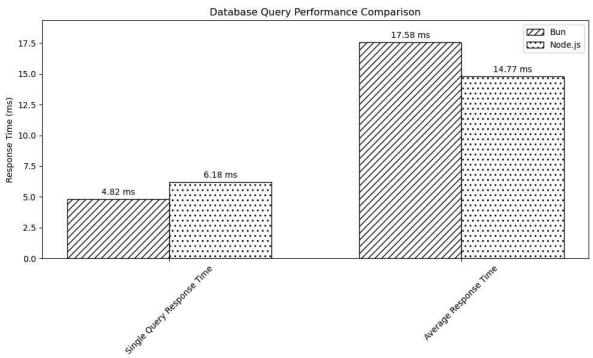

그림 10. 데이터베이스 질의 처리 성능 비교 그래프
Fig. 10. Database query processing performance comparison graph

5.4 정성평가

정량 평가를 진행한 결과 Bun은 트랜잭션 처리 속도에서 매우 우수한 결과를 보였다. 이는 대규모 트랜잭션을 처리할 때 Bun이 서버 부하를 적게 받고 더 신속하게 응답하는 능력이 뛰어나다는 것을 의미한다. 특히 티켓 발급과 같은 블록체인 기반 시스템에서는 이러한 신속한 처리 능력이 중요한데, Bun은 이를 효과적으로 수행할 수 있는 좋은 선택지이다.

Node.js는 트랜잭션 처리와 부하 테스트에서는 떨어지는 모습을 보였지만 데이터베이스 질의를 처리하는 데 있어 Bun보다 미세하지만 우수한 결과를 나타냈다.

3.2절에서 관련 연구를 통해 모든 부분에서 Bun이 Node.js보다 우수할 것이라 예측했다. 하지만 Node.js가 Bun보다 데이터베이이스 질의 처리 속도에 있어 우수한 결과를 나타냈다. 예측한 것과 다른 결과가 도출된 이유를 살펴보자면 출시 이후 오랜기간 최적화와 개선이 이루어지며 데이터베이스와 최적화된 라이브러리, 드라이브가 많이 개발되어온 Node.js와 달리 Bun은 아직 개발 초기 단계이기 때문에 이러한 부분에서 개발이 많이 이루어지지 않았다[18].

하지만 NFT 기반 티켓 양도 시스템과 같은 고성능 트랜잭션 처리가 필요한 시스템이라는 것을 고려한다면 Bun이 더 적합하다. 데이터베이스 질의 처리 성능은 현재 Node.js가 우수하지만, Bun도 최적화된 라이브러리나 드라이버의 도입 등을 통해 성능 개선이 가능하다.

Ⅵ. 결 론

이 논문에서는 NFT 기반 티켓 양도 시스템을 구축할 때 안정적인 서버를 서비스하기 위해 자바스크립트 런타임 환경을 비교하고 평가를 진행하여 위와 같은 시스템에서 어떤 자바스크립트 런타임 환경이 알맞은지 살펴봤다. 비교 항목은 트랜잭션 처리 속도와 데이터베이스 질의 처리 속도를 살펴봤다.

Bun은 단일 트랜잭션 처리에서 Node.js보다 2.41배 빠른 처리 속도를 보였고 부하 테스트의 평가 항목에서는 모두 Node.js보다 좋은 결과를 나타냈다. 하지만 데이터베이스 질의 처리 성능은 Node.js가 1.19배의 미세하게 높은 지표를 나타내며 초반 모든 부분에 있어 Bun이 우세할 것이라는 예상이 빗나갔다. 그 이유는 Node.js에 비해 Bun은 아직 개발 초기 단계이기 때문에 데이터베이스 관련하여 충분한 업데이트가 이뤄지지 않았기 때문이다. 하지만 트랜잭션 처리 속도와 고부하 환경에서의 안정성을 고려한다면 NFT 기반 티켓 시스템과 같은 블록체인 애플리케이션을 설계할 때 Node.js보다 Bun을 사용하는 것이 적합하다고 판단되지만, 데이터베이스 질의 처리 성능은 한계점으로 존재하고, 추후 Bun의 업데이트를 통해 최적화된 라이브러리나 드라이버를 도입한다면 충분히 보완 가능하다고 보여진다.

References

[1] S. Jo, "Visible 'Scalper Tickets' in Popular Concerts and Musicals, Can They Now Be Punished?", Civic News, Jun. 2023.

[2] C. Kim, "Clearly Seeing but Unable to Catch: The War with Scalpers That Only Catches Innocent Fans", Weekly Kyunghyang, May 2024.

[3] K. Nam and Y. Hwang, "A Study on the Issues and Improvements of Concert Ticket Resale", Hongik Law Review, Vol. 21, No. 1, pp. 293-314, 2020. https://doi.org/10.16960/jhlr.21.1.202002.293.

[4] A. Yoon, J. Yoon, S. Jin, and K. Lee, "Implementation of a Blockchain-based E-Ticket Purchase and Transfer System", Proc. of the KIEE EMECS, Yongpyeong, Korea, pp. 60-63, Jul. 2021.

[5] B. Lee, S. Moon, J. Yoo, J. Shin, and K. Kim, "A Study on Vulnerability Analysis Methodologies for Node.js", Proceedings of the Korean Information Processing Society Conference, Jeju, Korea, pp. 479-482, Oct. 2019.

https://doi.org/10.3745/PKIPS.y2019m10a.479.

[6] H. J. Oh, Z. Y. Heng, and R. Oh, "Development of Web Application for Railway Pattern Data Approach Using Node.js Modules", Proceedings of the Korean Society of Computer Information Conference, Daejeon, Korea, pp. 119-122, Jan. 2023.

[7] Bun Team, Bun Documentation, https://bun.sh/docs. [accessed: Oct. 8, 2024]

[8] A. Parravicini and R. Mueller, "The Cost of Speculation: Revisiting Overheads in the V8 JavaScript Engine", 2021 IEEE International Symposium on Workload Characterization (IISWC), Storrs, CT, USA, Nov. 2021. https://doi.org/10.1109/IISWC53511.2021.00013.

[9] H. Park, S. Kim, and S.-M. Moon, "Work-in-Progress: Advanced Ahead-of-Time Compilation for JavaScript Engine", 2017 International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), Seoul, Korea, pp. 1-2, Oct. 2017. https://doi.org/10.1145/3125501.3125512.

[10] W. K. Jung and S. M. Moon, "Call Optimization on Just-in-Time Compiler of V8 JavaScript Engine", Proc. of the Korean Information Science Society Conference, Gyeongju, Korea, pp. 135-138, Jun. 2011.

[11] S. Kim, "Recycling the Optimized Machine Codes Generated by JavaScript Engine", Master's Thesis, Department of Electrical and Computer Engineering, Graduate School of Seoul National University, pp. 1-28, Jul. 2017.

[12] H. Yoon and J. Chung, "A Case analysis of NFT digital art works", IIBC, Vol. 22, No. 5, pp. 55-61, Oct. 2022. https://doi.org/10.7236/JIIBC.2022.22.5.55.

[13] Bytes, "What Exactly Are the Differences Between JSC and V8?", https://bytes.dev/archives/109. [accessed: Oct. 10, 2024]

[14] J. Choi, "Design and Implementation of an NFT Ticket Issuance System to Prevent Scalping", Master's Thesis, Hanyang University Graduate School of Engineering, pp. 1-62, Aug. 2022.

[15] M. F. Ahmod, "JavaScript Runtime Performance Analysis: Node and Bun", Faculty of Information Technology and Communication Sciences (ITC) Master's Thesis, 2023.

[16] Y.-A. Min and D.-K. Lim, "Performance Analysis of Consensus Algorithm considering NFT Transaction Stability", IIBC, Vol. 22, No. 2, pp. 151-157, Apr. 2022. https://doi.org/10.7236/JIIBC.2022.22.2.151.

[17] S. Kang and H. Jung, "An Analysis on the Feasibility of Using Bun for Blockchain Web Applications", Proc. of the KIIT Fall Conference, Jeju, Korea, pp. 509-513, Nov. 2023.

[18] H. Choi and H. Kim, "A Blockchain-based NFT Blood Donation Certificate Management System", JKIIT, Vol. 22, No. 9, pp. 133-144, Sep. 2024. https://doi.org/10.14801/jkiit.2024.22.9.133.

저자소개

박 지 안 (Jian Park)

2024년 10월 : 국립군산대학교 소프트웨어학과(학사)
관심분야 : 블록체인, 웹 서버, 데이터베이스

정 동 원 (Dongwon Jeong)

1997년 2월 : 군산대학교 컴퓨터과학과(학사)
1999년 2월 : 충북대학교 전자계산학과(석사)
2004년 2월 : 고려대학교 컴퓨터학과(박사)
2005년 4월 ~ 현재 : 국립군산대학교 소프트웨어학과 교수
관심분야 : 데이터베이스, 시맨틱 서비스, 빅데이터,사물인터넷, 엣지컴퓨팅, 지능형 융합 서비스

정 현 준 (Hyunjun Jung)

2008년 3월 : 삼육대학교 컴퓨터과학과(학사)
2010년 3월 : 숭실대학교 컴퓨터학과(공학석사)
2010년 9월 : 고려대학교 컴퓨터·전파통신공학과(공학박사)
2017년 8월 ~ 2020년 8월 : 광주과학기술원 블록체인인터넷경제연구센터 연구원
2021년 3월 ~ 현재 : 국립군산대학교 소프트웨어학과 교수
관심분야 : 블록체인,데이터 사이언스,센서 네트워크,사물인터넷,머신러닝