JCSI 3 (2017) 6-10
Journal of Computer Sciences Institute

Wysłane: 2016-11-23
Przyjęte: 2016-11-30

Wykorzystanie Node.js w tworzeniu aplikacjach sterowanych zdarzeniami

Władysław Hrynczyszyn*, Małgorzata Plechawska-Wójcik
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. W niniejszej publikacji omówiona została nowa technika tworzenia oprogramowania – programowanie sterowane zdarzeniami. Została ona porównana z innymi popularnymi technikami tworzenia serwisów w celu ujawnienia słabych punktów oraz sprawdzeniu, w jakich obszarach aplikacji internetowych nadaje się do stosowania.

Słowa kluczowe: node.js; programowanie zdarzeniowe

*Autor do korespondencji.

Adres e-mail: vladyslav.gryn@gmail.com

Using of Node.js in creating application based on event-driven architecture

Władysław Hrynczyszyn*, Małgorzata Plechawska-Wójcik
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. This paper discusses a new programming method - event driven programming. This method is compared with other popular ways of implementing web services to find its weak points and discover in which areas of modern web applications it could be implemented.

Keywords: node.js; event-driven programming

*Corresponding author.

E-mail address: vladyslav.gryn@gmail.com

1. Wstęp

W związku z dynamicznym rozwojem Internetu oraz wykorzystywanych w nim technologii, aplikacje internetowe dostępne w sieci zaznały również dużych zmian. Wraz z ich rozwojem powstało wiele mechanizmów służących do tworzenia oprogramowania [1]. Jednym z takich mechanizmów jest programowanie sterowane zdarzeniami, na którym bazuje architektura platformy Node.js wykorzystywana w niniejszej publikacji [2].

Na dzień dzisiejszy istnieje wiele aplikacji internetowych wykorzystujących nową technikę tworzenia oprogramowania – programowanie sterowane zdarzeniami. Jej działanie polega na uruchomieniu programu obsługi zdarzeń w chwili wystąpienia odpowiedniego zdarzenia. Zdarzenia te wywoływane są asynchronicznie w stosunku do działania programu [3].

Platforma Node.js wykorzystywana jest w wielu produktach wytwarzanych i wspieranych przez znane firmy takie jak Yahoo czy Google. Osiągnięcie takiej popularności stało się możliwe dzięki jej wydajności oraz systemu paczek npm (*ang. node package manager)* [4]. Platforma ta stosowana jest zazwyczaj w serwisach z wysokim obciążeniem tzn. dużą liczbą odsłon strony. Na dzień dzisiejszy platforma programistyczna Node.js jest jedną z najbardziej popularnych platform tworzenia oprogramowania na runku i jej popularność wciąż rośnie [5].

W niniejszej publikacji przedstawione zostało porównanie dwóch technologii tworzenia aplikacji internetowych: PHP oraz Node.js. Każda z tych technologii ma swoje zastosowanie w różnych obszarach i bazuje na różnych paradygmatach programowania. Głównym celem jest zbadanie zachowania stworzonej aplikacji i w jej oparciu przeprowadzenie różnych testów, co pozwoli na ujawnienie słabych punktów, a następnie dokonanie analizy porównawczej. Po przeprowadzeniu analizy będzie można odpowiedzieć na pytanie, czy platforma Node.js oparta o technikę tworzenia oprogramowania „sterowanie przez zdarzenia" jest wydajna i czy nadaje się do serwisów z dużym przepływem danych oraz dużą liczbą użytkowników.

2. Opis aplikacji testowych

Aby móc dokonać analizy porównawczej powstały dwie małe testowe aplikacje internetowe napisane w popularnych technologiach tworzenia oprogramowania – PHP i Node.js [6]. Aplikacje zostały zaimplementowane w czystym kodzie PHP oraz JavaScript bez użycia żadnych frameworków, które łączą się ze wspólną bazą danych postawioną na PostgreSQL. Baza danych zawiera trzy tabele, które zostały wypełnione danymi. Po stworzeniu takich aplikacji przeprowadzone zostały różne testy.

Każda z zaprojektowanych aplikacji posiada listę funkcjonalności, które posłużą do przeprowadzenia badań.

Główną funkcjonalnością aplikacji jest posiadanie prostego interfejsu graficznego umożliwiającego wykonanie różnych operacji. Poniżej przedstawiona została lista dostępnych akcji.

Wyświetlanie danych – przykładowa aplikacja łączy się z bazą danych w celu pobrania oraz wyświetlenia różnej ilości informacji. Dla danego przypadku stworzone zostały trzy proste akcje:

- wyświetlanie listy użytkowników – stworzona została tabela zawierająca tysiąc rekordów;
- wyświetlanie stanowiska prac użytkowników – tabela zawiera 10000 rekordów;
- wyświetlanie listy artykułów użytkowników – w tym celu powstała tabela posiadająca 50000 rekordów.

Wyszukiwanie danych – akcja polega na wyszukaniu najbardziej popularnego artykułu

Filtrowanie danych – wyświetlanie użytkownika z pensja większą od pensji średniej wśród wszystkich użytkowników

Wstawianie danych – zadaniem akcji jest wstawienie stu rekordów do tabeli zawierającej informacje o użytkownikach

Operacje na tablicach – w tym przypadku wykorzystany został algorytm sortowania bąbelkowego, który przyjmuje tablice posiadającą 10000 losowych liczb i sortuje ją rosnąco

Operacje na plikach – aplikacja odwołuje się do API systemu operacyjnego w celu pobrania zawartości przykładowego folderu, a następnie wyświetlenie rozmiarów plików znajdujących się w tym katalogu. Dana operacja wywoływana została tysiąc razy

Operacja na tekście – zadaniem aplikacji jest wygenerowanie 10000 unikatowych ciągów znaków za pomocą różnych metod dostępnych w obu technologiach np. długość tekstu czy generowanie znaku.

3. Realizacja badania

W przedstawionej publikacji przeprowadzone zostały dwa eksperymenty, a następnie poddane analizie. Pierwszy eksperyment polegał na zbadaniu szybkości działania aplikacji napisanej w języku programowania PHP. Drugi eksperyment zawierał identyczne czynności jak w pierwszym przypadku, tylko że aplikacja została zaimplementowana przy użyciu platformy Node.js. Wszystkie testy wykonano na serwerze działającym lokalnie.

Każdy eksperyment polegał na wykonaniu dziewięciu różnych akcji. Pierwsze trzy akcje pobierały oraz wyświetlały różną ilość informacji z bazy danych. Kolejne trzy akcje polegały na wykonaniu różnych obliczeń oraz odwołań do funkcjonalności operacyjnego systemu. Ostatnie trzy akcje miały do czynienia z bazą danych - wykonane zostały takie działania jak: wyszukiwanie, filtrowanie oraz wstawianie danych do bazy.

Dla każdej akcji zostały wykonane trzy pomiary mierzenia czasu oczekiwania odpowiedzi z serwera, a następnie obliczona została wartość średnia. Warto zwrócić uwagę na przechowywanie danych w przeglądarkach – może to mieć duży wpływ na otrzymane wyniki, dlatego przy każdym teście czyszczona była pamięć podręczna przeglądarki.

4. Analiza badań

Porównane zostały dwie technologie tworzenia stron internetowych PHP oraz Node.js i przeprowadzono analizę otrzymanych wyników. Porównanie dotyczy takich aspektów aplikacji jak:

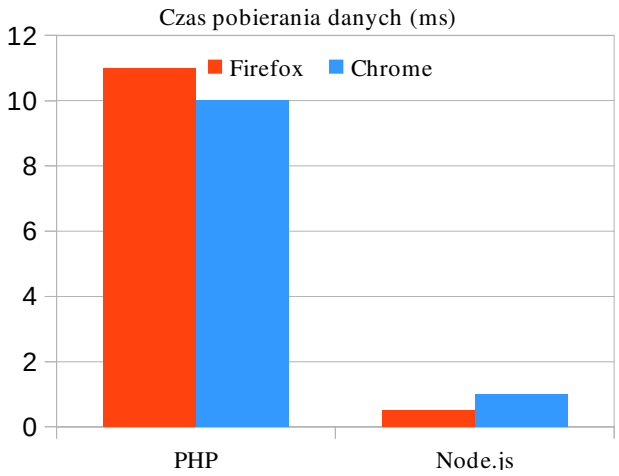
- szybkość pobierania żądania przez serwer, tworzenia odpowiedzi oraz przesłania danych w odpowiednim formacie;
- wykorzystanie zasobów fizycznej maszyny.

Celem przeprowadzenia analizy jest uzyskanie zalet oraz wad wykorzystanych technologii oraz sprawdzeniu, która z nich jest lepszy i przynosi większe korzyści w serwisach, w których głównym aspektem jest ich wydajność.

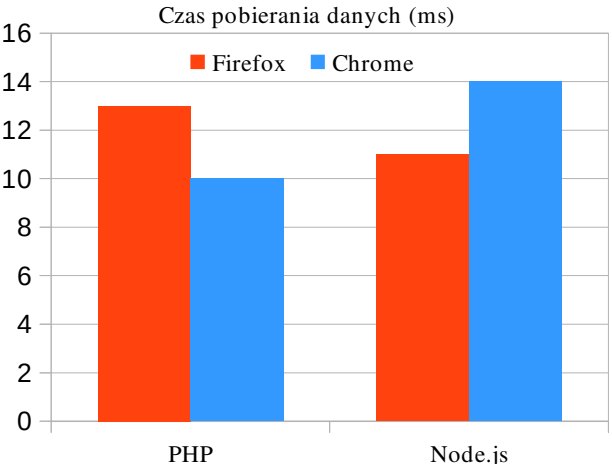
Poniżej został zaprezentowany szereg różnych testów. Testy polegały na zmierzeniu czasu odpowiedzi serwera. W celu przeprowadzenia analizy i otrzymania wyników, testy zostały podzielone na osiem etapów.

W pierwszym etapie przeprowadzono test wyświetlenia strony głównej, która zawiera listę dostępnych akcji. Jak widać na poniższym rysunku (Rys. 1) najlepszy czas osiągnięty został przez platformę Node.js. Czas potrzebny na wczytanie strony głównej jest w dziesięć razy mniejszy niż jest to w przypadku PHP.

W drugim etapie aplikacja testowa miała do czynienia z bazą danych. Testy zostały przeprowadzone przy użyciu trzech różnych akcji. Pierwszy test dotyczył wyświetlenia listy użytkowników (Rys. 2). W otrzymanych wynikach widać, że obie technologie są na podobnym poziomie, natomiast wydajność operacji związanych z pobieraniem danych z bazy przy użyciu platformy Node.js nie są zbyt szybkie i przy większym przepływie danych PHP sprawuje się dużo lepiej, co zostało zaobserwowane w kolejnych testach.



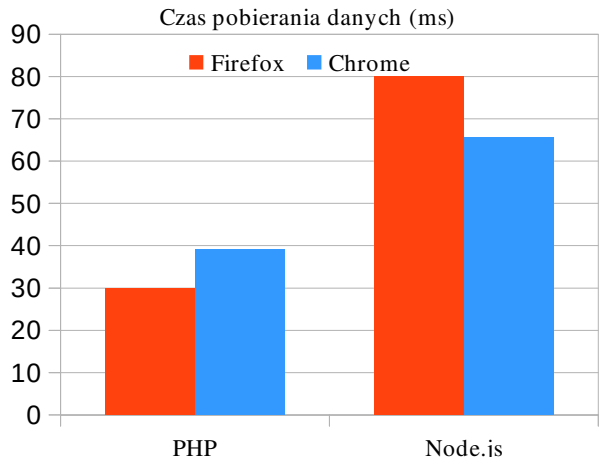
Rys. 1. Szybkość wczytywania strony głównej



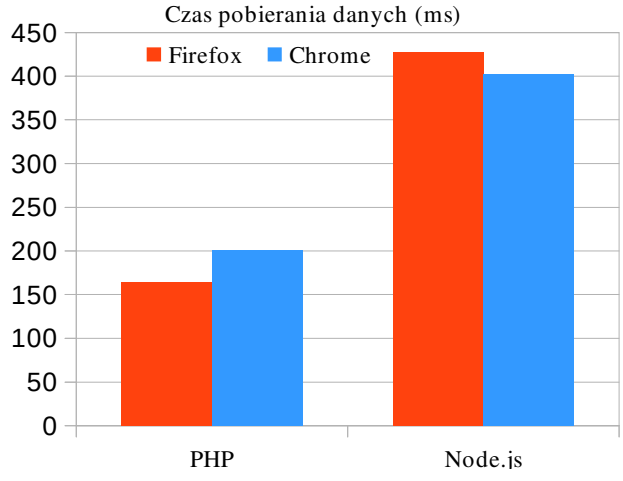
Rys. 2. Szybkość wczytywania listy użytkowników

W kolejnych testach można zauważyć, że przy większej ilości informacji pobieranych z bazy danych Node.js radzi sobie dużo gorzej niż PHP (Rys. 3 oraz Rys. 4). Dzieje się tak, ponieważ wykorzystywane biblioteki oraz połączenia z bazą danych mogą znacznie obniżyć wydajność platformy Node.js. Każda z nich mogła zostać zaimplementowana na różne sposoby lub przy tworzeniu przykładowej aplikacji mogły zostać popełnione różnego rodzaju błędy.

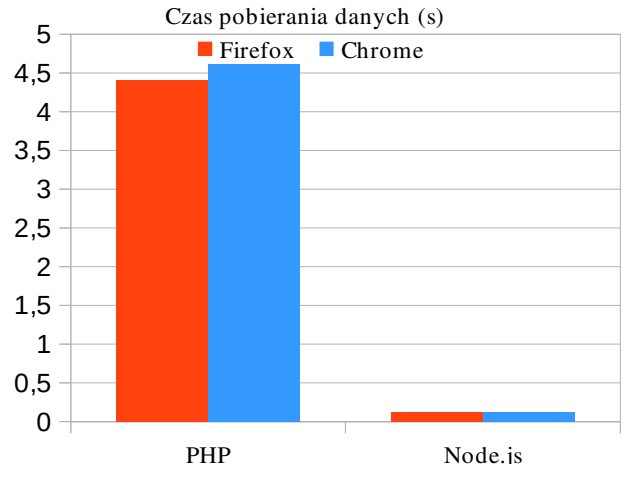
W trzecim etapie aplikacja testowa miała do czynienia z sortowaniem bąbelkowym. Taki typ testu potrafi mocno obciążyć procesor, co z kolei ma wpływ na wydajność aplikacji. Obserwując poniżej zamieszczony rysunek (Rys. 5) można stwierdzić, że Node.js potrafi dużo szybciej przetworzyć ciężkie operacje i sprawniej wykorzystuje moc obliczeniową procesora. Różnica w czasie jest ogromna – Node.js dokonał sortowania tablicy w 0,13 ms, a PHP prawie w 5s.



Rys. 3. Szybkość wczytywania listy stanowisk prac



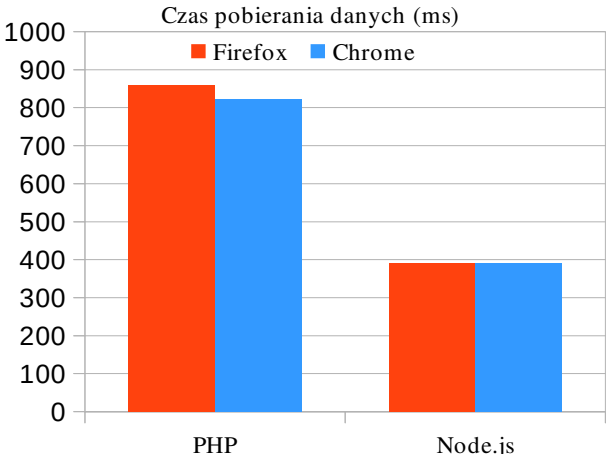
Rys. 4. Szybkość wczytywania listy artykułów



Rys. 5. Szybkość porządkowania tablicy liczb losowych

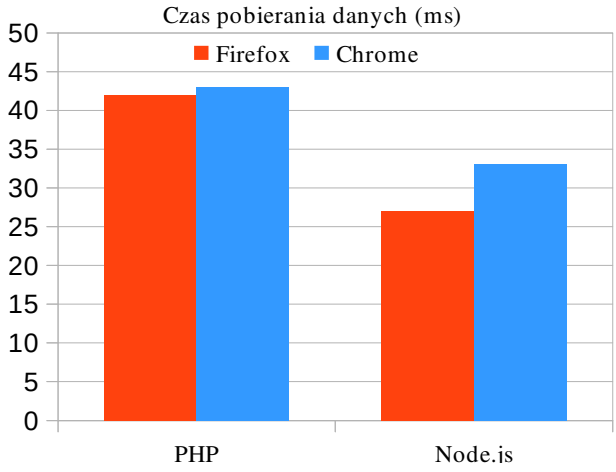
Czwarty etap polegał na odwołaniu się do udostępniających usług systemu operacyjnego – pobranie listy plików z przykładowego folderu, a następnie uzyskanie

rozmiaru dla każdego pliku. Obserwując rysunek (Rys. 6) można wnioskować, że platforma Node.js potrafi w szybki sposób obsłużyć odwołania do systemu plików, chociaż realizacja tej operacji w języku programowania PHP też jest na wysokim poziomie (różnica czasów wynosi około 400 ms na korzyść platformy Node.js).



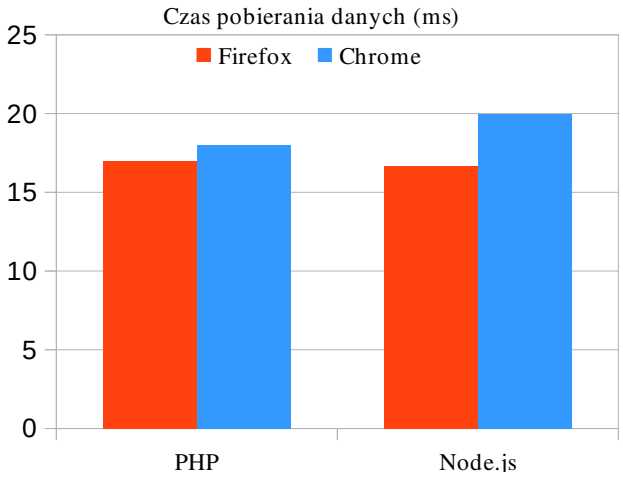
Rys. 6. Szybkość pobrania listy plików

W kolejnym piątym etapie badania obie technologie zostały przetestowane względem szybkości wykonania operacji na tekście. Rysunek 7 świadczy o tym, że aplikacje wygenerowały tekst w bardzo krótkim czasie i śmiało można stwierdzić, że działania takiego typu nie są specjalnie trudne dla obu technologii tworzenia stron internetowych.

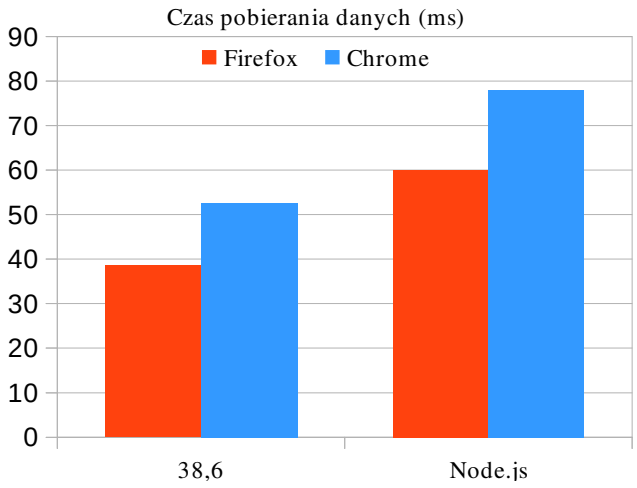


Rys. 7. Szybkość wykonania operacji na tekście

W kolejnych dwóch etapach aplikacje znowu miały do czynienia z bazą danych. Tym razem zadanie polegało na wyszukiwaniu danych – szukanie najpopularniejszego artykułu oraz filtrowanie danych pod kątem pensji użytkowników. Z otrzymanych rezultatów widać, że w przypadku mniejszej ilości danych pobieranych z bazy, czas odpowiedzi z serwera dla obu technologii jest podobny (Rys. 8). Ciekawa sytuacja występuje podczas pobierania większej ilości danych. Aplikacja oparta o platformę Node.js ma duży spadek wydajnościowy, co można zauważyć na rysunku 9. Jak wspomniałem wcześniej spowodowane to może być wieloma faktorami.



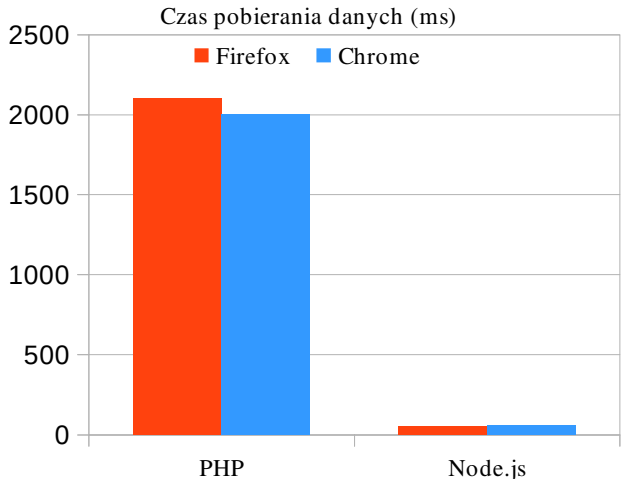
Rys. 8. Szybkość wyszukiwania artykułu



Rys. 9. Szybkość filtrowania użytkowników

Ostatni etap polegał na wstawianiu danych do tabeli. Zapytanie SQL wykonane zostało sto razy. Na przedstawionym poniżej rysunku widać, że czas osiągnięty przez platformę Node.js jest 52 razy mniejszy niż w przypadku PHP (Rys. 10). Dzięki asynchronicznemu modelowi Node.js nie czeka na wykonanie zapytania, tylko wykonuje następne [7]. Wszystkie takie operacje wykonywane są równolegle. W przypadku aplikacji napisanej w PHP wywołanie kolejnego zapytania nastąpi w chwili zakończenia poprzedniego [8]. Właśnie dlatego Node.js jest zalecany do stosowania w serwisach z dużą liczbą użytkowników.

Przy przeprowadzeniu wyżej opisanych testów ważnym aspektem było wykorzystanie dostępnych zasobów maszyny fizycznej. Zbyt duże ich zużycie potrafi spowolnić działania systemu, a czasami nawet do zatrzymania działania aplikacji.



Rys. 10. Szybkość wstawiania danych

Ważnym podzespołem serwera jest procesor. Szybkość działania operacji zależy od charakterystyk procesora oraz jego wykorzystania. Jak widać na zamieszczonej poniżej tabeli 1, serwer platformy Node.js przy 10000 jednoczesnych odwołaniach do strony głównej wykorzystuje tylko jeden z czterech dostępnych rdzeni i zużywa około 15% dostępnej mocy obliczeniowej procesora, natomiast serwer Apache przy identycznym teście wydajnościowym obciąża wszystkie cztery rdzenia procesora i wykorzystuje około 70% dostępnej mocy.

Kolejnym ważnym podzespołem serwera jest pamięć RAM. Obserwując tabele 1 można zauważyć, że platforma Node.js posiada nieco gorsze wyniki niż PHP. W przypadku technologii PHP wykorzystanie pamięci RAM w trakcie wykonywania testu nie zmieniło się, natomiast Node.js wykorzystał około 4% dodatkowej pamięci RAM. Pomimo to, ze względu na wydajność nowoczesnych serwerów (ilość dostępnej pamięci RAM), tak małe różnice nie będą miały dużego wpływu na wydajność aplikacji.

Tabela 1. Zużycie zasobów maszyny fizycznej

Serwer	Obciążenie procesora %				Wykorzystanie pamięci RAM %
	1 CPU	2 CPU	3 CPU	4 CPU	
Apache	68	65	68	65	35
Node.js	18	5	6	5	39

5. Wnioski

Platforma Node.js zdobywa coraz większą popularność wśród programistów. W ciągu ostatnich kilku lat platforma została zaakceptowana przez znane firmy i wykorzystywana w ich produktach [9].

Po analizie przeprowadzonych testów śmiało można stwierdzić, że Node.js jest naprawdę wydajny. W porównaniu z serwerem Apache, Node.js działa znacznie szybciej, ale dzięki tak dużej wydajności serwer platformy Node.js zużywa więcej zasobów komputera niż serwer Apache, a dokładnie pamięci RAM. Warto dodać, że Node.js nie jest prosty w nauczeniu, ponieważ architektura platformy bazuje na asynchronicznych działaniach i początkujący programista może popełnić wiele błędów, które spowodują duże zużycie zasobów maszyny fizycznej, a nawet doprowadzić zmniejszenia wydajności zaprojektowanej aplikacji.

Jeśli chodzi o wybór między tymi platformami to warto pamiętać, że Node.js nie będzie potrafił zastąpić PHP we wszystkich przypadkach, rozwiązać wszelkie problemy z architekturą aplikacji lub poprawić szybkość działania aplikacji przy dużym obciążeniu. Jeżeli wybór padł na użycie platformy Node.js w nowym projekcie lub na przepisanie starego, najpierw należy zastanowić się czy wybrana platforma będzie spełniać wszelkie wymagania projektu i czy będzie ona stanowić dobre rozwiązanie. Również w zależności od planowanych funkcjonalności aplikacji, wykorzystywanych bibliotek, połączeń z bazą danych, wydajność platformy Node.js może znacznie się różnić.

Podsumowując można stwierdzić, że architektura ma bardzo duże znaczenie dla wydajności całego systemu. Potencjał platformy Node.js i jej zastosowań dobrze pokazuje lista dostępnych modułów. Dostarczenie serwera z asynchronicznym I/O sprawia, że wielokrotne odpytywanie bazy nie prowadzi do tworzenia kolejnych operacji, ponieważ wykonywane są one równolegle [10]. W czasie oczekiwania na odpowiedź z bazy danych lub innych usług, serwer platformy Node.js potrafi bez żadnego problemu przyjmować kolejne połączenia i generować dla nich odpowiedzi. Dzięki temu Node.js potrafi ekonomicznie wykorzystywać zasoby komputera, a przede wszystkim daje duży przyrost wydajności. Warto również dodać, że używanie platformy Node.js w projekcie zależy od wiele czynników: specyfikacji projektu, doświadczenia programisty, a użycie platformy nie zawsze rozwiąże wszelkie problemy.

Literatura

[1] D. M. Simmonds, The Programming Paradigm Evolution, 2012
[2] C. Gackenheimer, Understanding Node.js, 2013
[3] C. J. Ihrig, Pro Node.js for Developers, 2013
[4] B. Syed, Beginning Node.js, 2014
[5] A. Mardan, Practical Node.js Building Real-World Scalable Web Apps, 2014
[6] K. Lei, Y. Ma, Z. Tan, Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js, 2014
[7] D. Vohra, Using Node.js, 2015
[8] S. Nakajima, An Architecture of Dynamically Adaptive PHP-based Web Applications, 2011
[9] S. Tilkov, S. Vinoski, Node.js: Using JavaScript to Build High-Performance Network Programs, 2010
[10] A. Ojamaa, K. Düüna, Assessing the security of Node.js platform, 2012