بهینه‌سازی اجرا و پاسخ صفحات وب در فضای ابری با روش‌های پیش‌پردازش، مطالعهٔ موردی سامانه‌های وارنیش و انجینکس

سنیه دیلمی و یعقوب فرجامی*

گروه مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه قم، قم، ایران

چکیده

سرعت پاسخ صفحات وب یکی از ضرورت‌های دنیای فناوری اطلاعات است. در سال‌های اخیر شرکت‌های مشهور از قبیل گوگل و دانشمندان علوم رایانه تمرکز خود را روی سرعت‌بخشیدن به وب قرار دادند و تلاش کردند تا راهی برای سریع‌ترکردن وب بیابند. دستاوردهایی از قبیل گوگل پیج اسپید، انجینکس و وارنیش نتیجه این پژوهش‌ها بوده است. درهمین‌اواخر روش‌ها و ابزارهای مؤثر و موفقی برای افزایش سرعت بارگذاری صفحات وب ارائه شده که بیش‌تر به دو رهیافت افزایش سرعت در سمت کاربر و افزایش سرعت در سمت سرور تقسیم می‌شود. پراکسی معکوس به‌عنوان مؤثرترین روش افزایش سرعت در سمت سرور است. در این مقاله ضمن معرفی پراکسی معکوس، کارایی سامانه‌های وب سرور چهارگانه آپاچی + وارنیش، انجینکس، انجینکس + وارنیش و آپاچی را، با دو نوع محتوای پویا و ایستا، از منظر سرعت پاسخ صفحات وب به‌عنوان معیار سنجش بررسی کرده‌ایم. نخست این‌که نتایج به‌دست‌آمده نشان می‌دهند، استفاده از پراکسی معکوس موجب افزایش سرعت پاسخ می‌شود؛ دوم این‌که این افزایش سرعت نه‌تنها به وب سرور بلکه به نوع محتوای صفحات وب و تعداد تقاضاهای تکراری در صفحات وب مرتبط است. در پایان یک رتبه‌بندی برای انتخاب وب‌سرور و پراکسی معکوس مناسب برای محتوای وب ایستا یا چندرسانه‌ای و محتوای وب پویا یا پردازشی ارائه شده است.

واژگان کلیدی: سرعت پاسخ، پراکسی معکوس، انجینکس، آپاچی، وارنیش

Optimizing Web programs Response in Cloud Using Pre-processing, Case study Nginx, Varnish

Saniyeh Deylami & Yaghoub Farjami*

Department of Computer and Information Technology, University of Qom, Qom, Iran

Abstract

The response speed of Web pages is one of the necessities of information technology. In recent years, renowned companies such as Google and computer scientists focused on speeding up the web. Achievements such as Google Pagespeed, Nginx and varnish are the result of these researches. In Customer to Customer(C2C) business systems, such as chat systems, and in Business to Customer(B2C) systems, such as online stores and banks, the power and speed of the system's response to the high volume of visitors are very effective in customer satisfaction and the efficiency of the business system. Increasing the speed of web pages from the origin of the advent of this technology, used from known and proven methods such as preprocessing, cookie, Ajax, cache and so on, to speed up the implementation of Internet applications, but it still needs to increase the speed of running and operating systems under the web.

* نویسندهٔ عهده‌دار مکاتبات

* Corresponding author

Recently, successful and effective methods and tools devised to increase the loading speed of Web pages, which consist mainly two approaches, increasing the speed on the client-side user and increasing the speed of the server-side.

Research and technology on the performance and speed of Web technology on server side are divided into two categories of content enhancements, such as the Google Page Speed tool and Web server performance improvements such as Reverse Proxies. Reverse proxy is the most effective way to increase the speed on the server-side. Web server performance is measured by various metrics such as process load, memory usage and response speed to requests. Reverse proxy technology has been implemented in the Varna and Engineer systems. Implementing the reverse proxy in Varna has focused on caching processing content and on the engineering to cache static content.

Our goal is to evaluate the performance of these two systems as reverse proxies to improve the response speed and loading of web pages in two types of dynamic (processing) and static (multimedia) content and provide a framework for the appropriate selection of a reverse proxy on web servers.

In this paper, we introduce reverse proxy and analyze the performance of the four web servers, namely apache + varnish, nginx, nginx + varnish and apache, with both static and dynamic content, in terms of response speed of web pages as a measure of performance. First, our results show that, using a reverse proxy response speed is increased. Second, the resulted speed up is related not only to web server type but also to the content type of web pages requested repeatedly. Finally, a ranking is provided which helps to select the appropriate web server and reverse proxy when the web content type is static (multimedia) or dynamic (processed).

Keywords: reverse proxy, varnish, nginx, accelerated web, response time.

۱- مقدمه

در وب‌سایت‌های تجاری، سرعت اجرا و پاسخ سریع صفحات وب ضرورتی حیاتی جهت رضایت مشتریان و در کلیه سامانه‌های تحت وب افزایش سرعت موجب کارآمدی آنها است. در سامانه‌های تجاری مشتری به مشتری[۱]، مانند وبگاه‌های بلاگ و سامانه‌های چت، و تجارت به مشتری[۲]، مانند فروشگاه‌های اینترنتی و بانک‌ها، با توجه به حجم بالای مراجعه‌کنندگان و بازدیدکنندگان که هریک عملیات متنوعی را انجام می‌دهند، توان و سرعت پاسخ سامانه امری بسیار تأثیرگذار در رضایت مشتریان و کارآمدی سامانهٔ تجاری است.

از لحاظ سابقهٔ تاریخی، افزایش سرعت صفحات وب از ابتدای ظهور این فناوری با روش‌های شناخته‌شده و سابقه‌داری از قبیل پیش‌پردازش[۳]، کوکی[۴]، آژاکس[۵]، کش‌کردن[۶] و ... برای سرعت‌بخشیدن به اجرای برنامه‌های اینترنتی استفاده شده‌اند؛ ولی همچنان نیاز به بالابردن سرعت اجرا و بهره‌وری سامانه‌های تحت وب احساس می‌شود. قابل توجه است که همه این روش‌ها تأکید به بهبود پردازش در سمت کاربر و مرورگر بوده و در روش‌های افزایش سرعت پاسخ صفحات وب در سمت سرور تا سال‌های اخیر دستاورد چشم‌گیری وجود نداشته است که با ظهور روش پراکسی معکوس[۷] در سمت سرور در سال ۲۰۱۰ اوضاع بسیار تغییر کرد [1,2,8].

در مورد سامانه‌های تجارت الکترونیک با پایه تجارت به مشتری مثل بانک‌های اینترنتی، سامانه‌های پرداخت و تسویه‌حساب اینترنتی و ... نیز به‌دلیل تعامل هم‌زمان با تعداد زیادی از مشتریان، سرعت اجرا و پاسخ‌دهی بلادرنگ اهمیت ویژه‌ای پیدا می‌کند.

کارایی خدمات تجارت الکترونیک فقط به موارد بالا محدود نمی‌شوند و اهمیت سرعت و پاسخ‌دهی بلادرنگ در سامانه‌های دیگری مثل شبکه‌های اجتماعی، موتورهای جستجو، سامانه‌های مشارکتی تبلیغات بازرگانی، سامانه‌های مشارکتی چندرسانه‌ای، سامانه‌های ویدئوکنفرانس، سامانه‌های تلفن اینترنتی، سامانه‌های چت، سامانه‌های ای آر پی[۸]، سی آرام[۹] نیز، که بر مبنای فناوری وب هستند، ضروری می‌نماید [1,2].

پژوهش‌ها و فناوری‌های کارایی و سرعت فناوری وب در سمت سرور خود به دو دستهٔ بهبود محتوا، مانند ابزار گوگل پیج‌اسپید[۱۰] و بهبود عملکرد سرور وب مانند پراکسی معکوس تقسیم می‌شوند. عملکرد سرور وب با معیارهای مختلفی از قبیل بار پردازشی، حافظهٔ اشغال‌شده و سرعت پاسخ به

[1] Customer to Customer(C2C)
[2] Business to Customer(B2C)
[3] Pre Compile
[4] Cookie
[5] Ajax
[6] Caching

[7] Reverse proxy
[8] Enterprise Resource Planning(ERP)
[9] Customer Relationship Management(CRM)
[10] Google pagespeed

درخواست‌ها سنجیده می‌شود. فناوری پراکسی معکوس در دو سامانهٔ وارنیش[1] و انجینکس[2] پیاده‌سازی شده است. پیاده‌سازی پراکسی معکوس در وارنیش به کش‌کردن محتوای پردازشی و در انجینکس به کش‌کردن محتوای ایستا معطوف شده است.

هدف ما بررسی کارایی این دو سامانه به‌عنوان پراکسی معکوس در بهبود سرعت پاسخ و بارگذاری صفحات وب در دو نوع محتوای پویا (پردازشی) و ایستا (چندرسانه‌ای) و ارائه یک چارچوب برای انتخاب مناسب یک پراکسی معکوس در وب‌سرورها است.

۲-پیشینه پژوهش

با توجه به اهمیت کارایی فناوری وب، سرمایه‌گذاری و پژوهش‌های زیادی در این زمینه صورت گرفته است که هرکدام از آنها به جنبه‌های مختلفی از کارایی توجه کرده‌اند. در سال ۲۰۱۰ آقای کمپ با انتقاد از ره‌یافت سنتی به مفهوم پیچیدگی الگوریتم‌ها به تحلیل جدید و مؤثری از پراکسی معکوس پرداخت و کارایی آن را در سامانهٔ وارنیش بررسی کرد[8]. آقای برگر و همکارانش در سال ۲۰۱۷ با به‌کارگیری روش زنجیره مارکوف به روشی برای اولویت‌دهی اشیای ذخیره‌شده، براساس حجم آنها، در کش سمت سرور دست یافتند و نشان دادند که نتایج آنها در شبکه توزیع محتوای[3] چندرسانه‌ای در بهبود عملکرد بسیار مؤثر است [25]. آقای توبیاس در سال ۲۰۱۴ با بررسی تعداد تقاضای هم‌زمان روی وب‌سرور با قابلیت پراکسی معکوس، دو سامانهٔ وارنیش و انجینکس را بررسی کردند و نشان دادند، از نظر درگیری حافظه و بار عملیاتی پردازنده، عملکرد انجینکس در مدیریت تعداد بالای تقاضای هم‌زمان از وارنیش بهتر است [13].

در یک پژوهش مفصل در سال ۲۰۱۶ آقای تکلیمانوت به موضوع کارایی وب سرورها برای پخش ویدئو پرداخته و یک سامانهٔ سخت‌افزاری به‌نام اوربیت[4] را معرفی می‌کند. تفاوت اصلی این پژوهش با دیگران این است که نخست این‌که به روش‌های سخت‌افزاری بهبود عملکرد وب‌سرور و دوم این‌که به‌صورت انحصاری به کارایی وب‌سرور پخش ویدئو توجه شده است. در این پژوهش کارایی اوربیت و وارنیش (یک سامانهٔ سخت‌افزاری در مقابل یک سامانهٔ نرم‌افزاری) در اجرای پراکسی معکوس با محتوای ویدئو مقایسه و اوربیت با کارایی بالاتر معرفی شده است [15]. آقای چگان در مقالات متعددی به مسئله سرعت پاسخ صفحات پرداخته، و برعکس روش پراکسی معکوس، به خوشه‌بندی کاربرها با استفاده از داده‌کاوی و مدل‌های مارکوف پرداخته و نشان داده روش پیشنهادی موجب افزایش سرعت پاسخ صفحات وب می‌شود؛ ولی مقایسه‌ای با ابزارهای پراکسی معکوس انجام نداده است [27,28]. آقای کومار و همکاران در مقاله ۲۰۱۴ خود به روش‌های توزیع بار تقاضا در یک خوشه از سرورها در فضای ابری پرداختند. آنان با ارائه یک الگوریتم ترکیبی برای توزیع بار در خوشه سرورها نشان دادند که روش پیشنهادی موجب بهبود عملکرد در پاسخ به تقاضای کاربر و بارگذاری صفحات وب می‌شود؛ ولی در پژوهش خود مقایسه‌ای با پراکسی معکوس انجام ندادند [16].

موضوع سرعت پاسخ به تقاضای کاربر در سامانه‌های پخش ویدئو و تلویزیون اینترنتی به واسطه نوع محتوا که از صفحات معمولی وب متفاوت است و به‌طورمعمول تعداد تقاضای هم‌زمان گاهی به چندین میلیون می‌رسد، مورد پژوهش‌های متعددی بوده است. در مقاله سال ۲۰۱۶ آقای نوگریا و همکاران به ارائه الگوریتمی پرداختند که در آن اولویت اشیای کش سمت سرور تلویزیون اینترنتی براساس آخرین تقاضاها، بیشترین تقاضاها و کمترین تقاضاهای اخیر تنظیم می‌شود. در این مقاله نسبت تقاضای کاربران از کش به کل تقاضاها (معیار برخورد کش به تقاضاهای جاری) به‌عنوان یک معیار جدید کارایی معرفی شده و اثر اجرای الگوریتم کش سمت سرور پیشنهادی در بهبود این معیار اثبات شده است [21]. لوشل و اشمیت در سال ۲۰۱۳ به مسئله کارایی سرورهای نقشه و خدمات اطلاعات جغرافیایی پرداختند. در این پژوهش روش‌های مختلفی برای کش نقشه‌های جغرافیایی سمت سرور بررسی، و به‌خوبی نشان داده‌اند، روش‌های متداول کش کارایی لازم را ندارند؛ بلکه روش‌های بهبود محتوا می‌تواند کارایی سرور را سرعت پاسخ به تقاضاهای همزمان فراهم کند [20].

در سال ۲۰۱۴ کتابی با عنوان "بهبود عملکرد سایت‌های مگنتو[5]" نوشته شد که به‌صورت خاص به سامانهٔ مدیریت محتوای مگنتو پرداخته است. در این کتاب ابتدا یک سری سامانه‌های سخت‌افزاری برای بهبود عملکرد مگنتو معرفی شده و با سه وب‌سرور آپاچی[6]، لایت تی پی دی[7] و انجینکس شرایط کارایی آزموده شده است. از این رو، می‌بایست، ابتدا وب‌سرورها را نصب کرده و پس از بررسی

[1] varnish
[2] nginx
[3] Content Delivery Network (CDN)
[4] Orbit Streaming Server
[5] Magento
[6] Apache Http Server
[7] Lighttpd

چگونگی بهینه‌سازی آن‌ها، در نهایت به وب‌سرور بهینه دست پیدا کند. در ادامه سازوکارهای مختلف کش‌کردن را بررسی کرده تا درخواست‌های کاربران را با استفاده از آن ذخیره کند. در این راستا، اپلیکیشن وارنیش معرفی شده و به شرح عملکرد آن پرداخته است[14].

مقالات و کتاب‌های بسیارِ دیگری نیز در زمینهٔ بهینه‌سازی و بهبود عملکرد وب‌سرورها با سامانه‌های محتوای مختلف و سایت‌های مختلف به‌صورت جزیی‌تر به‌عنوان‌مثال کاربرد وارنیش در سامانه‌های ابری موجود است،[17-19].

در پژوهش‌های انجام‌شده تاکنون مقایسه کارایی روش‌های پراکسی معکوس در بهبود سرعت پاسخ از نظر محتوای ایستا یا چندرسانه‌ای و محتوای پویا یا پردازشی انجام نشده است که در این مقاله به آن می‌پردازیم.

۳- ادبیات پژوهش

۳-۱- وب‌سرور آپاچی (اچ.تی.تی.پی سرور)

وب‌سرور آپاچی به زبان سی[1] نوشته شده است، و از زبان‌های سمت سرور پرل[2] و پی اچ پی پشتیبانی می‌کند و بر روی سیستم‌عامل‌های مختلف قابل‌اجرا است. یک برنامهٔ کدباز آزاد[3] است که در سرورهای وب برای اداره‌کردن درخواست‌ها و تقاضاهای وب و منابع به کار می‌رود و در حال حاضر نزدیک به ۴۹ ٪ بازار سرورهای وب جهان را به خود اختصاص داده است. همچنین سیستم‌عامل مک آن را به‌عنوان سرور وب اصلی خود برگزیده است. سرور وب آپاچی برای میزبانی هر دو نوع وب ایستا و وب پویا مناسب است [5,23,24].

آپاچی امکانات ویژه‌ای دارد که متداول‌ترین استفاده از ویژگی‌های این برنامه دات اچ تی اکسس[4] است که طراحان حرفه‌ای در محیط لینوکس از آن بهره می‌گیرند. برای نمونه، زمانی که بخواهند نخستین صفحه در سایت ویژه‌ای باشد با یک دستور در آن پرونده این امر ممکن می‌شود و یا زمانی که صاحب سایت مایل نیست که فایل‌های موجود در سرور وی توسط دیگران دزدیده شود و بخواهد که مانع از پیوند مستقیم آن‌ها شود، آپاچی کمک می‌کند تا به خواستشان برسند. زمانی‌که برنامه‌نویس بخواهد محل واقعی صفحات دیده نشود نیز این برنامه مورداستفاده قرار می‌گیرد [5].

[1] C
[2] Perl
[3] Free Open Source
[4] .htaccess

بزرگ‌ترین مشکل وب سرور آپاچی محدودیت اتصالات هم‌زمان ده‌هزارتایی و هم‌چنین میزان مصرف بالای حافظه است که به مشکل C10k معروف است؛ این مشکل باعث شده است که با آمدن انجینکس محبوبیت آپاچی کاهش پیدا کند[3].

۳-۲- پراکسی معکوس

اندیشهٔ اثربخشی بسیار بالای پراکسی معکوس در افزایش سرعت پاسخ وب‌سرورها را آقای کمپ در سال ۲۰۱۰ مطرح کرد. ایده اصلی آن توسعه مفهوم کارآمد کش[5] از سمت کاربر به سمت سرور است. خود آقای کمپ قبلاً در سال ۲۰۰۶ یک پیاده‌سازی متن باز از پراکسی معکوس با نام وارنیش ارائه کرده و در مقاله خود کارایی آن را اثبات کرد [8]. پراکسی معکوس یک نوع پراکسی است که بر سر راه وب‌سرورها قرارگرفته و تمامی درخواست‌ها را دریافت می‌کند. این نوع پراکسی با بررسی تمامی درخواست‌ها، در صورت لزوم آن‌ها را برای سرور ارسال می‌کند و درصورتی‌که خودپاسخی برای آن داشته و درقبل آن پاسخ را در خود پیش‌پردازش و ذخیره کرده باشد، درخواست را برای سرور ارسال نکرده و خودش پاسخ را می‌فرستد. این امر باعث می‌شود که سرور درگیر درخواست‌های مکرر و تکراری نشود و درنتیجه سرعت پاسخ به درخواست‌ها به‌شدت افزایش می‌یابد [7,19,22].

۳-۳- وب‌سرور انجینکس

انجینکس یک سرور کدباز پراکسی معکوس برای پروتکل‌های اچ.تی.تی.پی، اچ.تی.تی.پی.اس[6]، اس.ام.تی.پی[7]، پاپ-۳[8] و آی مپ[9] است که به‌خوبی به‌عنوان متعادل‌کنندهٔ بارگذاری، کش اچ.تی.تی.پی و یک وب‌سرور عمل می‌کند. پروژهٔ انجینکس با هدف افزایش کارایی، هم‌زمانی بالا و مصرف حافظهٔ کم شروع به کار کرد. این وب‌سرور رایگان است؛ حجم پایین و کارایی بسیار بالایی دارد و تحت لیسانس بی.اس.دی[10] منتشر می‌شود. یکی از بزرگ‌ترین مزیت‌های این وب‌سرور، پشتیبانی بسیار عالی از فایل‌های ایستا است.

انجینکس سرعت پاسخ بسیار بالایی دارد و در بازدیدهای هم‌زمان پرتعداد بسیار عالی عمل می‌کند. ازجمله قابلیت‌های انجینکس سازگاری این وب‌سرور با بیش‌تر توزیع‌های لینوکس هست. زمانی که بخواهید نسخهٔ انجینکس را ارتقا دهید، به‌صورت در پرواز[11] این کار را انجام می‌دهد و

[5] Cache
[6] HTTPS (Hypertext Transfer Protocol Secure)
[7] SMTP
[8] POP3
[9] IMAP
[10] BSD
[11] On the fly

این در مورد پروژه‌هایی که سایت نباید زمان توقف[1] داشته باشد، خیلی اهمیت دارد [2,3,6].

انجینکس برای مدیریت درخواست‌ها یا همان اتصالات از روش منحصر به خودش استفاده می‌کند. در این سرور برای هر درخواست یک نخ[2] جدید درست می‌شود و این در حالی است که سیستمی که حتی مخزن نخ[3] دارد هم نخ را تا زمانی حفظ می‌کند که در آن لحظه درخواست جدید بیاید و انجینکس از نخ‌های بیکار استفاده می‌کند. این روش استفاده از نخ‌ها در انجینکس بسیار کارآمد است. به‌طورکلی مدیریت باز و بسته‌شدن نخ‌ها در وب‌سرورهایی مثل آپاچی باعث شده که این وب‌سرور در اتصالات هم‌زمان پرتعداد، در پاسخ به درخواست‌های بارگذاری صفحات وب دچار کندی و اختلال شود. انجینکس در سال‌های ۲۰۱۳ و ۲۰۱۶ برنده جایزه جهانی نرم‌افزارهای آزاد شد [3,6].

۴-۳- شتاب‌دهنده‌ای به نام وارنیش

وارنیش یک شتاب‌دهندهٔ اچ.تی.تی.پی است که برای وب‌سایت‌های پویا با محتوای سنگین طراحی شده است. برخلاف تسریع‌دهنده‌های وب دیگر نظیر اسکوییدِ[4] که به‌عنوان کش سمت کاربر عمل می‌کند، یا آپاچی و انجینکس که سرورهای مقدماتی هستند، وارنیش به‌عنوان یک شتاب‌دهندهٔ اچ.تی.تی.پی طراحی شده است. برخلاف دیگر سرورهای پروکسی که اغلب اف.تی.پی[5]، اس.ام.تی.پی و دیگر پروتکل‌های شبکه را پشتیبانی می‌کنند، وارنیش به‌صورت اختصاصی روی اچ.تی.تی.پی متمرکز شده است. وارنیش در سال‌های ۲۰۱۲ و ۲۰۱۳ برنده جایزه جهانی نرم‌افزارهای آزاد شد [4,7,12,13].

۱-۴-۳- معماری

وارنیش داده را در حافظهٔ مجازی ذخیره می‌کند و سیستم‌عامل را از عمل تصمیم‌گیری در مورد این‌که چه چیزی در حافظه وجود دارد و چه صفحه‌ای باید از دیسک خارج شود، رها می‌کند. فایدهٔ این کار آن است که از به‌وجودآمدن موقعیت‌هایی اجتناب می‌کند، که سیستم‌عامل عمل کش‌کردن داده را شروع می‌کند. درحالی‌که آن داده توسط برنامه به دیسک انتقال پیداکرده است؛ به‌علاوه وارنیش به‌شدت به نخ کشیده است، با هر ارتباط کاربر به‌وسیلهٔ یک نخ جداگانه اداره می‌شود. هنگامی‌که حد مشخصی روی تعدادی از نخ‌های فعال در دسترس است، ارتباطات ورودی در صف سرریز قرار می‌گیرند. وقتی این صف به ارتباطات ورودی محدودهٔ مشخص خودش دسترسی داشته باشد، رد خواهد کرد.

سازوکار پیکربندی اصلی، زبان پیکربندی وارنیش[6] است، یک زبان مشخصهٔ دامنه[7] که برای نوشتن تله[8] هایی استفاده می‌شود که در نقاط بحرانی سازمان‌دهی هر درخواست صدا زده می‌شوند. برنامه‌ریزی و تصمیم این‌که کدام اشیا در کش نگهداری و یا چگونه از کش حذف شوند به کد وی.سی.ال واگذار می‌شود، که این ویژگی وارنیش را نسبت به شتاب‌دهنده‌های دیگر اچ.تی.تی.پی منعطف‌تر و سازگارتر می‌سازد. وقتی یک اسکریپت[9] وی.سی.ال بارگذاری شده و در کامپایلر سیستم‌عامل سرور، کامپایل و به‌صورت مستقیم بارگذاری می‌شود، می‌تواند بدون راه‌اندازی، دوباره پیکربندی شود.

تعدادی از پارامترهای زمان اجرا، کنترل‌کردن چیزهایی نظیر بیشینه و کمینهٔ تعداد نخ‌های کارکننده[10]، تایم اوت[11] های جدی و غیره است. یک واسط مدیریت خط فرمان[12]ها به این پارامترها اجازه می‌دهد که تعدیل شوند، و اسکریپت وی.سی.ال‌های جدید کامپایل شوند، بارگذاری شده و بدون راه‌اندازی دوبارهٔ شتاب‌دهنده فعال شوند.

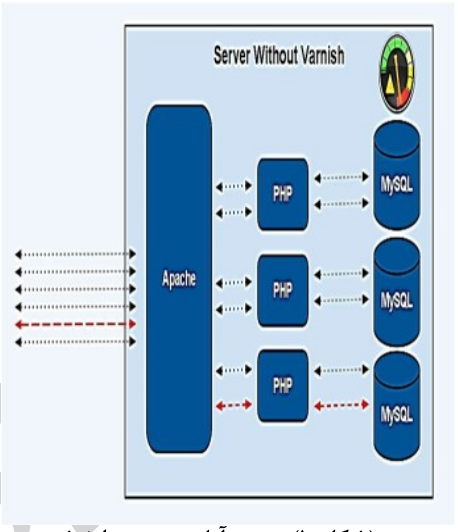
به‌منظور کاهش تعداد سامانه فراخوانی‌ها در مسیر سریع به یک کمینه، دادهٔ لاگ در حافظهٔ مشترک ذخیره می‌شود، و عمل بازبینی، فیلترکردن، فرمت‌کردن و نوشتن دادهٔ لاگ به دیسک به برنامه جدا نمایندگی می‌دهد [4,7].

اکنون با دو شکل نمادین زیر، به‌صورت جزئی معماری وارنیش را شرح می‌دهیم. شکل (۱) سرور آپاچی را بدون حضور تسریع‌دهنده وارنیش نشان می‌دهد. در این حالت آپاچی به‌صورت نظیر به نظیر با پی.اچ.پی[13] در ارتباط دوطرفه است و پی.اچ.پی نیز داده و اطلاعات را از مای.اس.کیو.ال[14] می‌گیرد و به آن اطلاعات می‌دهد و در کل با هم در ارتباط دوطرفه هستند.

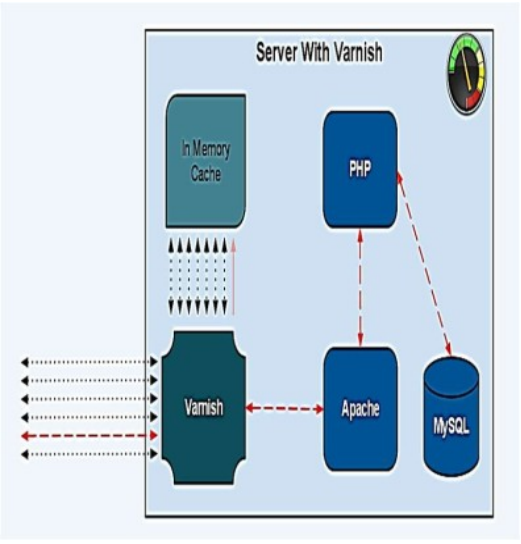
شکل (۲) سرور آپاچی را به‌همراه تسریع‌دهنده وارنیش نشان می‌دهد. همان‌طور که ملاحظه می‌کنید، وارنیش با یک حافظه کش در ارتباط است. درخواست‌ها ابتدا

[1] Down Time
[2] Thread
[3] Thread Pool
[4] Proxy e-mail
[5] FTP
[6] Varnish Configuration Language(VCL)
[7] Domain-Specific Language(DSL)
[8] Hook
[9] Script
[10] Worker Thread
[11] Timeout
[12] Command Line
[13] PHP
[14] My SQL

به وارنیش می‌رود و سپس وارنیش آن‌ها را با حافظه کش رد و بدل می‌کند، به عبارت دیگر مقداری از درخواست‌ها و حتی پاسخ درخواست‌ها را کش می‌کند تا در مواقعی که درخواست‌های مکرر یکسان داریم، بدون مراجعه به سرور اصلی پاسخ توسط وارنیش داده شود. وارنیش با آپاچی رابطه دوسویه دارد، آپاچی هم به‌وسیله یک ارتباط دوطرفه به پی.اچ.پی متصل است و پی.اچ.پی هم با مای.اس.کیو.ال که منبع داده و اطلاعات است، رابطهٔ دوطرفه برقرار می‌کند.



(شکل-۱): سرور آپاچی بدون وارنیش
(Figure-1): Apache server without varnish



(شکل-۲): سرور آپاچی به همراه وارنیش
(Figure-2): Apache server with varnish

ادعای مخترع وارنیش، آقای هنینگ این است که کارایی این سیستم به‌خوبی به‌کارگیری نخ پی[1]های سیستم عامل است؛ درحالی که وارنیش طراحی شد تا رقابت بین نخ‌ها به کمینه کاهش یابد [7,8]. بنابراین پیشنهاد عمومی اجرای وارنیش روی محیط‌های بر پایه لینوکس یا یونیکس است [7,4].

در ادامهٔ فصل با مفاهیم وب‌سرورها و دو نوع سرور با اهمیت آپاچی و انجینکس بحث می‌شود؛ سپس توضیح مفصلی درمورد وارنیش، ویژگی‌ها، کارایی و معماری آن داده و درنهایت به‌صورت مختصر به نرم‌افزارهای مبتنی بر وب پرداخته می‌شود.

۲-۴-۳- شما این کار را اشتباه انجام می‌دهید [8]

در سال ۲۰۱۰ آقای پل هنینگ کمپ[2]، از دانشمندان مشهور علوم رایانه، که برنامه‌نویس سیستم عامل مشهور بی.اس.دی آزاد و جزو پنج توسعه‌دهندهٔ اصلی آن بود، مقاله‌ای انقلابی با نگرشی جدید به محاسبات مربوط به پیچیدگی الگوریتم‌ها نوشت[8]. در آن مقاله تهاجم شدیدی به نحوه تحلیل الگوریتم‌ها کرد و افزود: «باور می‌کنید اگر من ادعا کنم الگوریتمی که به مدت ۴۶ سال به‌عنوان الگوریتم بهینه در کتاب‌ها بوده است و با جزئیات فراوان توسط فرد نابغه‌ای مثل دونالد کنوت[3] تحلیل شده باشد و در تمام رشته‌های علوم رایانه جهان تدریس شده باشد، می‌تواند به مقدار ده بار سریع‌تر بهینه شود؟ عنوان مقاله همه‌چیز را روشن می‌کند: *شما این کار را اشتباه انجام می‌دهید*».

کمپ در این مقاله پیشنهادهایی را برای افزایش سرعت الگوریتم‌ها ارائه کرده است که به‌اختصار شرح می‌دهیم. به‌طورمعمول و در تمامی سیستم‌های تحت وب، داده‌ها در سمت کاربر کش و کوکی[4] می‌شدند، از اسکریپت[5] و فناوری آژاکس استفاده می‌کردند و در سمت سرور هیچ عمل خاصی برای بهینه سازی سرعت انجام نمی‌شد. پیشنهاد این بود که ما می‌توانیم عملیات کش، کوکی، اسکریپت و تمامی عملیاتی را که سمت کاربر انجام می‌شد، در سمت سرور انجام دهیم و با این کار سرعت بالاتری در پاسخ به درخواست‌های سرور داشته باشیم.

ایده اصلی کمپ در بازنگری به مفهوم پیچیدگی بدین‌صورت است که هنگام محاسبهٔ پیچیدگی الگوریتم‌ها و استفاده از مقادیر $O(n)$، $O(n^2)$ و غیره، n همیشه به سمت بی‌نهایت میل می‌کند؛ در حالی که با نگاه واقع‌گرایانه و روزمره، n به سمت بی‌نهایت میل نمی‌کند. در کاربردهای وب n بین ده‌هزار تا بیست‌هزار است و به سمت بی‌نهایت میل نمی‌کند. او ثابت کرد، الگوریتم‌هایی وجود دارند که با n زیر

[1] PThread

[2] Poul-Henning Kamp
[3] Knuth
[4] Cookie
[5] Script

ده‌هزار خیلی بهتر از الگوریتم‌های به‌صورت نظری بهینه کار می‌کنند. این نظریات باعث انتقادهای شدیدی از طرف دانشگاه‌های مختلف ازجمله دانشگاه استنفورد[1] و برکلی[2] به کمپ شد [7,8].

کمپ درنهایت برای اثبات نظریه‌اش، سامانهٔ موردنظرش را که وارنیش نام‌گذاری شد، طراحی و پیاده‌سازی کرد و بازدهی محتوای ایستا را تا سیصد برابر افزایش داد. او برای این پیاده‌سازی هیچ تغییری در معماری سامانه‌های وب ایجاد نکرد و تنها با انجام پراکسی معکوس، عملیات کش در سمت کاربر را به سمت سرور انتقال داد و برای اینکه بتواند با وب‌سرورها کار کند، سامانه‌اش را طوری طراحی کرد که با هر وب‌سروری ازجمله آپاچی و انجینکس و تامکت[3] سازگار باشد. این کار، انقلاب عظیمی در بازدهی وب‌سرورها بود و باعث شد کار کمپ هنینگ در سال‌های ۲۰۱۲ و ۲۰۱۳ در بنیاد بی.او.اس.آی.ای[4] برندهٔ جایزه شود.

نخستین کاربر وارنیش یک روزنامهٔ بزرگ نروژی به نام وی.جی[5] بود که دوازده ماشین در حال اجرای اسکوییدِ[6] خود را با سه ماشین در حال اجرای وارنیش جایگزین کرد. ماشین‌های اسکویید صددرصد اشغال بودند؛ درحالی‌که ماشین‌های وارنیش نود درصد پردازندهٔ آزاد داشتند.

وارنیش به‌صورت هوشمندانه از حافظه مجازی بهره‌برداری می‌کند. سیصد گیگابایت ذخیرهٔ پشتیبان، حافظه روی یک ماشین با کمتر از شانزده گیگابایت رم[7] (به‌عنوان فضای نشان‌دهی) نگاشت می‌شود که واقعاً چشم‌گیر است.

یک تکلیف ویژه برای وارنیش، دورانداختن اشیا از درون کش است، وقتی‌که مدت زندگی مجازی آن‌ها تمام‌شده است. این کار برای ساختمان داده‌هایی انجام می‌شود که می‌توانند کوچک‌ترین شیء کلیددار را از مجموعه کلی دریافت کنند [8].

۴- پیاده‌سازی و آزمون

در این پژوهش روی چهار نوع وب‌سرور، عمل پیاده‌سازی و آزمون زمان بارگذاری (آزمون زمان پاسخ) صفحات وب را انجام دادیم که به‌ترتیب زیر می‌باشند: آپاچی، انجینکس، آپاچی + وارنیش و انجینکس + وارنیش. همچنین از شش نوع وب‌سایت و سامانهٔ مدیریت محتوا بانام‌های جوملا[8]، وردپرس[9] ،دروپال[10]، لایونس[11]، کار[12] و رایتل[13] استفاده کردیم. به این صورت که تمامی وب‌سایت‌ها و سامانه‌های مدیریت محتوا را روی تمامی وب‌سرورها نصب کردیم که درمجموع ۲۴ مورد نصب شد.

برای دقت بیشتر در کار از سه سایت آزمون سرعت وب پیج تست[14]، جی تی متریکس[15] و پینگ دام[16] استفاده کردیم. برای ثبت زمان‌های بارگذاری صفحات در مرورگرها از افزونه پیج لود تایم[17] و پیج بنچمارک[18] در مرورگر کروم[19] و اپ تله متری[20] در مرورگر فایرفاکس[21] استفاده کردیم. تمام سامانه‌های ردگیری به خارج از صفحات را غیر فعال کردیم. تک‌تک وب‌سرورهای نصب‌شده را روی هر سه سایت از نظر سرعت پاسخ آزمودیم؛ که درنهایت سه جدول شامل نتایج زمان پاسخ وب‌سرورها در این سه وب‌سایت به‌دست آمد [11][10][9]. میانگین نهایی زمان‌های پاسخ در جدول زیرنمودارها ارائه شده است.

در جدول (۱) ستون نخست فهرست وب‌سرورها، ستون دوم فهرست محتواها و ستون سوم سامانه‌های سنجش کارآمدی است.

(جدول-۱): فهرست تعداد حالات مورد بررسی

(Table-1): The number of cases rewiewed

وب سرور	محتوای وب	سنجش کارآمدی
Apache	Drupal پردازشی	WebpageTest
Nginx	Joomla پردازشی	GT-Metrix
Varnish + Apache	WordPress پردازشی	Pingdom
Varnish + Nginx	Lyoness ایستا	Page benchmark
	Car.ir ایستا	Page load time
	Rightel ایستا	App.telemetry

تعداد کل حالت‌های موردبررسی به شرح زیر است:

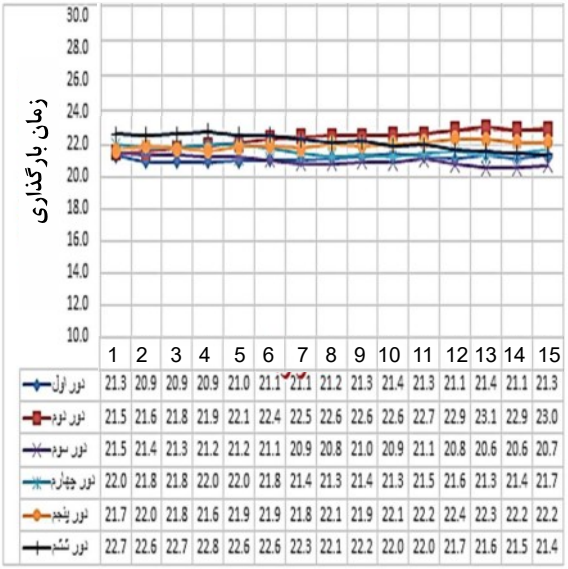
[1] Stanford
[2] Berkley
[3] Tomcat
[4] BOSSIE
[5] VG
[6] Squid
[7] RAM
[8] Joomla
[9] Wordpress
[10] Drupal
[11] Lyoness
[12] Car
[13] Rightel
[14] WEBPAGETEST
[15] GTmetrix
[16] Pingdom
[17] Page load time
[18] Page benchmark
[19] Chrome
[20] App.telemetry
[21] firefox

(۴ نوع وب سرور)×(۶ نوع محتوا)×(۶ نوع سنجش) ×(۶ تکرار فراخوانی)=۸۶۴ مورد

پارامترها عبارتند از زمان بارگذاری کامل صفحه وب و بهبود زمان پاسخ با تکرار فراخوانی. تمام سرورها روی لینوکس اوبونتو ۱۴,۰۴ با رم چهار گیگابایت و با پردازنده چهار هسته‌ای با سرعت ۳,۴ گیگاهرتز اجرا شده‌اند و در سمت کاربر نیز از سیستم‌عامل ویندوز و مرورگر فایرفاکس استفاده شده است؛ در آغاز هر آزمون تمام سوابق فراخوانی‌های قبلی پاک شده‌اند تا در نتیجه آزمون‌ها اختلال احتمالی ایجاد نشود.

محتوای وب مورد آزمون به دو صورتِ پویا و چندرسانه‌ای ایستا در نظر گرفته شده است. برای محتوای پویا سه نوع سامانهٔ مدیریت محتوا، جوملا،دروپال،وردپرس، در نظر گرفته‌ایم و در هر مورد نتایج میانگین زمان بارگذاری این سه نوع سامانه محتوای پویا ثبت شده است. همچنین برای محتوای چندرسانه‌ای ایستا محتوای سه سایت مختلف با مقادیر زیاد تصاویر و ویدئو در نظر گرفته شده‌اند و میانگین زمان بارگذاری آنها درج شده است. نتایج مربوط به آزمون‌ها به تفکیک در نمودارها و جداول زیر آمده است.

ابتدا به نتایج میانگین زمان بارگذاری محتوای ایستا در سرور آپاچی می‌پردازیم. همان‌طور که در نمودار (۱) دیده می‌شود، به‌دلیل سنگین‌بودن محتوای بارگذاری‌شده زمان اولیه حدود ۲۲ ثانیه و با تکرار حدود ۱۵ بار و در شش دور نتیجه به‌طورتقریبی ثابت باقی مانده است. به‌دلیل عدم وجود سامانهٔ کش و پیش‌پردازش محتوایی عدم بهبود کارایی در سامانهٔ آپاچی به‌طورکامل قابل انتظار است.

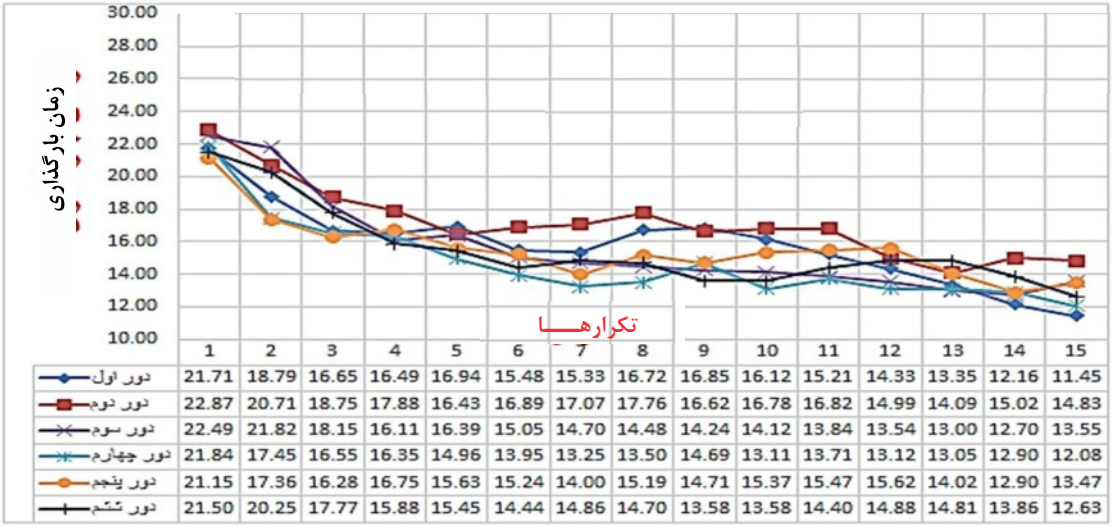


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
دور اول	21.3	20.9	20.9	20.9	21.0	21.1	27.1	21.2	21.3	21.4	21.3	21.1	21.4	21.1	21.3
دور دوم	21.5	21.6	21.8	21.9	22.1	22.4	22.5	22.6	22.6	22.6	22.7	22.9	23.1	22.9	23.0
دور سوم	21.5	21.4	21.3	21.2	21.2	21.1	20.9	20.8	21.0	20.9	21.1	20.8	20.6	20.6	20.7
دور چهارم	22.0	21.8	21.8	22.0	22.0	21.8	21.4	21.3	21.4	21.3	21.5	21.6	21.3	21.4	21.7
دور پنجم	21.7	22.0	21.8	21.6	21.9	21.9	21.8	22.1	21.9	22.1	22.2	22.4	22.3	22.2	22.2
دور ششم	22.7	22.6	22.7	22.8	22.6	22.6	22.3	22.1	22.2	22.0	22.0	21.7	21.6	21.5	21.4

(نمودار-۱): آزمون زمان بارگذاری محتوای چندرسانه‌ای با سرور آپاچی

(Chart-1): Test the loading time of the multimedia content with the Apache server

نمودار (۲) نتایج آزمون محتوای چندرسانه‌ای ایستا با سرور آپاچی + وارنیش است. از نتایج مشخص است که به واسطه وجود پراکسی معکوس (وارنیش) در سمت سرور و قابلیت کش بعضی از عناصر محتوای وب که به‌صورت مکرر مورد نیاز است، با تکرار تقاضاها، وارنیش یاد می‌گیرد که بعضی از محتواهای چندرسانه‌ای را از قبیل تصویر و ویدئو سریع‌تر بارگذاری کند. به‌همین دلیل نمودار زیر نشان می‌دهد که وجود وارنیش در کنار آپاچی باعث می‌شود زمان بارگذاری از حدود ۲۲ ثانیه اولیه به حدود ۱۲ ثانیه برسد که نتیجه قابل قبول و مورد انتظاری است.

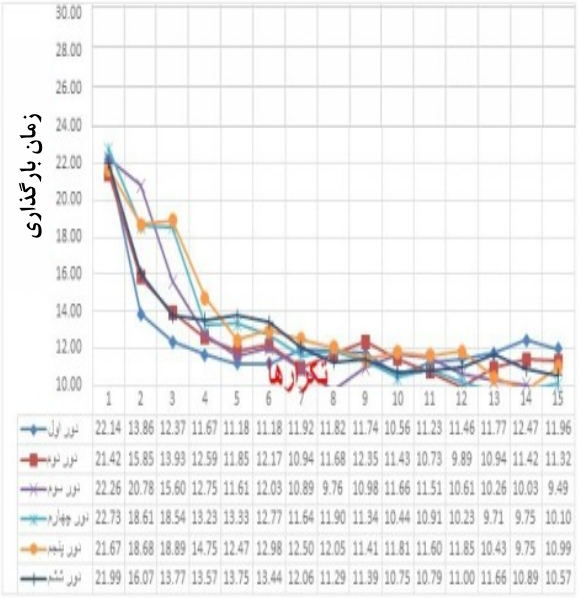


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
دور اول	21.71	18.79	16.65	16.49	16.94	15.48	15.33	16.72	16.85	16.12	15.21	14.33	13.35	12.16	11.45
دور دوم	22.87	20.71	18.75	17.88	16.43	16.89	17.07	17.76	16.62	16.78	16.82	14.99	14.09	15.02	14.83
دور سوم	22.49	21.82	18.15	16.11	16.39	15.05	14.70	14.48	14.24	14.12	13.84	13.54	13.00	12.70	13.55
دور چهارم	21.84	17.45	16.55	16.35	14.96	13.95	13.25	13.50	14.69	13.11	13.71	13.12	13.05	12.90	12.08
دور پنجم	21.15	17.36	16.28	16.75	15.63	15.24	14.00	15.19	14.71	15.37	15.47	15.62	14.02	12.90	13.47
دور ششم	21.50	20.25	17.77	15.88	15.45	14.44	14.86	14.70	13.58	13.58	14.40	14.88	14.81	13.86	12.63

(نمودار-۲): آزمون زمان بارگذاری محتوای چندرسانه ای با سرور آپاچی + وارنیش

(Chart-2): The test of time to load multimedia content with Apache + Varnish server

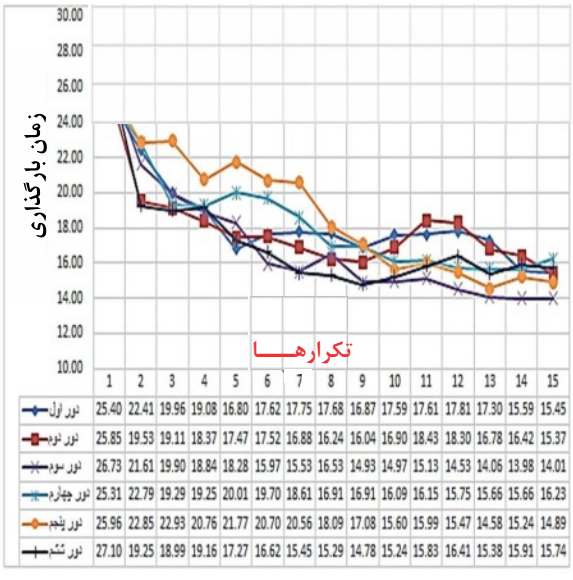
نمودار (۳) نتایج آزمون محتوای چندرسانه‌ای ایستا با سرور انجینکس است. این سرور دارای یک سامانهٔ کش است؛ لذا بدیهی است که دارای بهبود عملکرد در طی تکرارها خواهد بود که در جدول زیر مشخص است و زمان بارگذاری از حدود ۲۲ ثانیه اولیه به حدود ۱۰ ثانیه می‌رسد که نتیجه‌ای فوق‌العاده مناسب است.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
نور اول	22.14	13.86	12.37	11.67	11.18	11.18	11.92	11.82	11.74	10.56	11.23	11.46	11.77	12.47	11.96
نور دوم	21.42	15.85	13.93	12.59	11.85	12.17	10.94	11.68	12.35	11.43	10.73	9.89	10.94	11.42	11.32
نور سوم	22.26	20.78	15.60	12.75	11.61	12.03	10.89	9.76	10.98	11.66	11.51	10.61	10.26	10.03	9.49
نور چهارم	22.73	18.61	18.54	13.23	13.33	12.77	11.64	11.90	11.34	10.44	10.91	10.23	9.71	9.75	10.10
نور پنجم	21.67	18.68	18.89	14.75	12.47	12.98	12.50	12.05	11.41	11.81	11.60	11.85	10.43	9.75	10.99
نور ششم	21.99	16.07	13.77	13.57	13.75	13.44	12.06	11.29	11.39	10.75	10.79	11.00	11.66	10.89	10.57

(نمودار-۳): آزمون زمان بارگذاری محتوای چندرسانه‌ای با انجینکس

(Chart-3): The test of time to load multimedia content with Nginx



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
نور اول	25.40	22.41	19.96	19.08	16.80	17.62	17.75	17.68	16.87	17.59	17.61	17.81	17.30	15.59	15.45
نور دوم	25.85	19.53	19.11	18.37	17.47	17.52	16.88	16.24	16.04	16.90	18.43	18.30	16.78	16.42	15.37
نور سوم	26.73	21.61	19.90	18.84	18.28	15.97	15.53	16.53	14.93	14.97	15.13	14.53	14.06	13.98	14.01
نور چهارم	25.31	22.79	19.29	19.25	20.01	19.70	18.61	16.91	16.91	16.09	16.15	15.75	15.66	15.66	16.23
نور پنجم	25.96	22.85	22.93	20.76	21.77	20.70	20.56	18.09	17.08	15.60	15.99	15.47	14.58	15.24	14.89
نور ششم	27.10	19.25	18.99	19.16	17.27	16.62	15.45	15.29	14.78	15.24	15.83	16.41	15.38	15.91	15.74

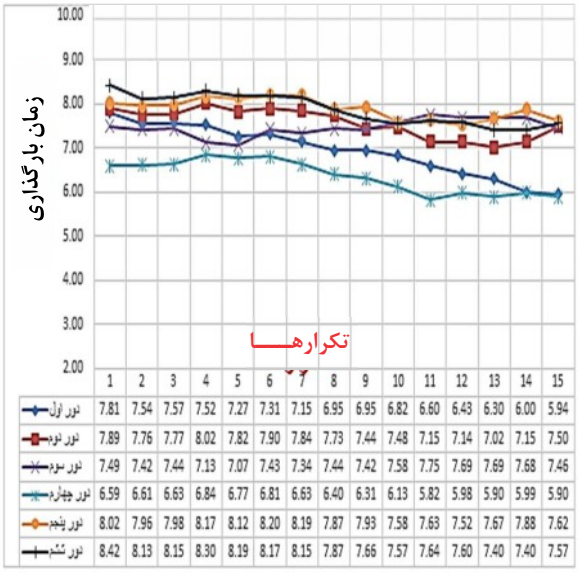
(نمودار-۴): آزمون زمان بارگذاری محتوای چندرسانه‌ای با سرور انجینکس+وارنیش

(Chart-4): The test of time to load multimedia content with Nginx + varnish server

در نمودار (۴) آزمون بارگذاری محتوای ایستا در سرور انجینکس + وارنیش انجام شد. همان‌طور که دیده می‌شود، اضافه‌کردن وارنیش به انجینکس نه‌تنها باعث بهبود عملکرد انجینکس نشد، بلکه زمان پاسخ‌دهی اولیه را تا حدود ۲۶ ثانیه افزایش داد و منحنی یادگیری وارنیش را هم بدتر کرد و زمان بهبود یافته نهایی در پانزده تکرار به حدود پانزده ثانیه رسید؛ که از دو نتیجه قبلی بدتر است. البته با تکرار بیشتر نتایج، کمی بهتر می‌شوند؛ ولی به کیفیت انجینکس و آپاچی + وارنیش نمی‌رسد.

حال به آزمون بارگذاری محتواهای پویا می‌رسیم. در این نوع محتواها در سمت سرور برنامه‌هایی به زبان‌های پی.اچ.پی، جاوا، پایتون، دات نت و مشابه آنها نوشته شده است. به‌طورمعمول براساس تقاضای کاربر پردازش و اجرای کد برنامه‌نویسی شده و فراخوانی و پرس و جو از پایگاه داده سمت سرور انجام می‌شود. نتیجه پردازش به‌عنوان محتوای نهایی به کاربر ارسال می‌شود. ما در آزمون‌ها از سه سامانهٔ مدیریت محتوای جوملا، دروپال و وردپرس استفاده کردیم و البته در تمامی موارد مقدار کمی نیز محتوای تصویر و آیکن وجود دارد. زمان ثبت‌شده میانگین زمان‌های ثبت‌شده در این سه سامانهٔ مدیریت محتوا است که همگی نیز به زبان پی.اچ.پی نوشته شده‌اند [3]. بدیهی است که نسبت به محتوای چندرسانه‌ای ایستا که شامل فایل‌های حجیم بودند، در حالت پویا که مقدار کمی فایل تصویر وجود دارد، سریع‌تر باشد که در نتایج هم مشخص است. سؤال اصلی در حالت محتوای پویا این است که آیا سامانه‌هایِ سرورِ مختلفِ موردِ آزمون، قادر هستند علاوه‌بر بهبود عملکرد به واسط کش نسبت به پردازش و ذخیره‌سازی نتایج پردازش‌شده برای تسریع در پاسخ به کاربر استفاده کنند یا خیر. از توصیف فنی آپاچی می‌دانیم که قابلیت کش ندارد. در مقابل انجینکس قابلیت کش دارد، ولی قابلیت ذخیره پیش‌پردازش‌شدهٔ کدهای سمت سرور را ندارد. وارنیش دارای قابلیت کش و ذخیره‌سازی پیش‌پردازش است [3,14,7].

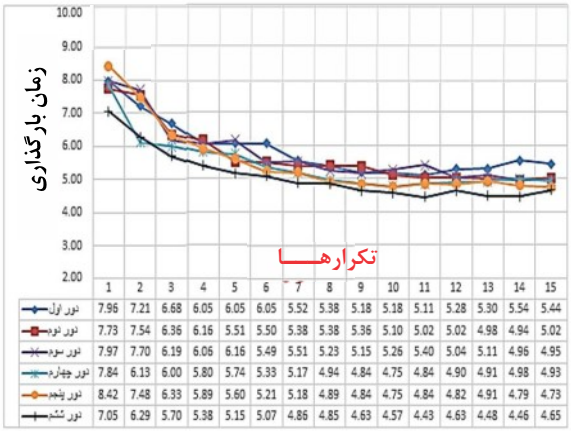
ابتدا نتایج آزمون سرور آپاچی با محتوای پویا را در نمودار (۵) ملاحظه می‌کنیم. در این نتایج همان‌طور که انتظار می‌رفت، نخست این‌که زمان بارگذاری نسبت به محتوای سنگین چند رسانه‌ای کمتر است. دوم این‌که با تکرار هیچ بهبودی در زمان پاسخ حاصل نشده است.



(نمودار-۵): آزمون زمان بارگذاری محتوای پویا با سرور آپاچی

(Chart-5): The test of time to load dynamic content with Apache server

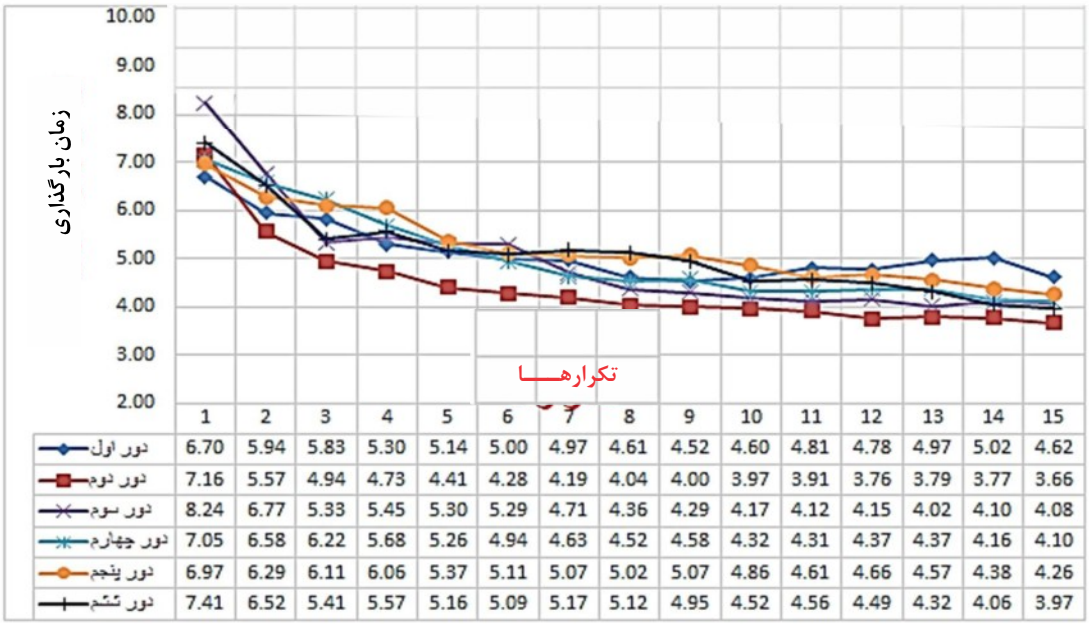
در ادامه آزمون را با سرور انجینکس و محتوای پویا اجرا می‌کنیم. نتایج در نمودار (۶) نشان می‌دهد که به واسطه قابلیت کش، رفته‌رفته با تکرار فراخوانی سرور عملکرد بهتری پیدا می‌کند؛ به‌نحوی که از زمان حدود هفت ثانیه پس از پانزده تکرار به زمان حدود چهار ثانیه می‌رسد. این نتیجه نشان می‌دهد که سرور انجینکس نسبت به آپاچی دارای قابلیت بهبود به واسطه کش است.



(نمودار-۶): آزمون زمان بارگذاری محتوای پویا با سرور انجینکس

(Chart-6): The test of time to load dynamic content with Nginx server

حال به سرور آپاچی + وارنیش می‌پردازیم. در این حالت به واسطه وجود قابلیت کش و قابلیت ذخیره‌سازی کدهای پردازش‌شده (پیش‌پردازش) انتظار داریم که عملکرد بهتری ملاحظه کنیم که در نمودار (۷) مشخص است. مقایسهٔ نتایج در سرور انجینکس و سرور آپاچی + وارنیش نشان می‌دهد که سرور آپاچی + وارنیش عملکرد بهتری دارد و در محتوای پویا بهتر از انجینکس عمل می‌کند.

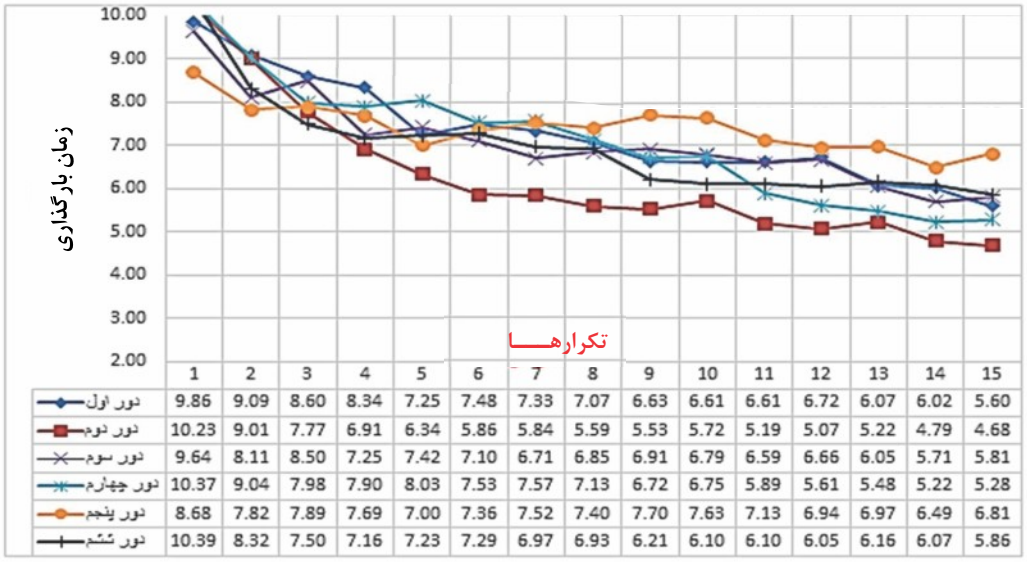


(نمودار-۷): آزمون زمان بارگذاری محتوای پویا با سرور آپاچی + وارنیش

(Chart-7): The test of time to load dynamic content with Apache + Varnish server

مورد نهایی آزمون ما در مورد سرور انجینکس + وارنیش است. تصور ما بر این بود که ممکن است، استفادهٔ همزمان از انجینکس که دارای قابلیت کش است و وارنیش که دارای قابلیت کش و ذخیره پیش‌پردازش است، عملکرد بسیار

بهتر شود. نتایج نمودار (۸) نشان می‌دهد که عملکرد انجینکس + وارنیش از عملکرد انجینکس ضعیف‌تر است که البته این نتیجه هم در مورد محتوای چندرسانه‌ای ایستا و هم در مورد محتوای پویا وجود دارد. سرور انجینکس + وارنیش از بابت زمان اولیه حتی از سرور آپاچی هم ضعیف‌تر است و از نظر زمان نهایی هم به حدود پنج ثانیه می‌رسد که از انجینکس ضعیف‌تر است.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
دور اول	9.86	9.09	8.60	8.34	7.25	7.48	7.33	7.07	6.63	6.61	6.61	6.72	6.07	6.02	5.60
دور دوم	10.23	9.01	7.77	6.91	6.34	5.86	5.84	5.59	5.53	5.72	5.19	5.07	5.22	4.79	4.68
دور سوم	9.64	8.11	8.50	7.25	7.42	7.10	6.71	6.85	6.91	6.79	6.59	6.66	6.05	5.71	5.81
دور چهارم	10.37	9.04	7.98	7.90	8.03	7.53	7.57	7.13	6.72	6.75	5.89	5.61	5.48	5.22	5.28
دور پنجم	8.68	7.82	7.89	7.69	7.00	7.36	7.52	7.40	7.70	7.63	7.13	6.94	6.97	6.49	6.81
دور ششم	10.39	8.32	7.50	7.16	7.23	7.29	6.97	6.93	6.21	6.10	6.10	6.05	6.16	6.07	5.86

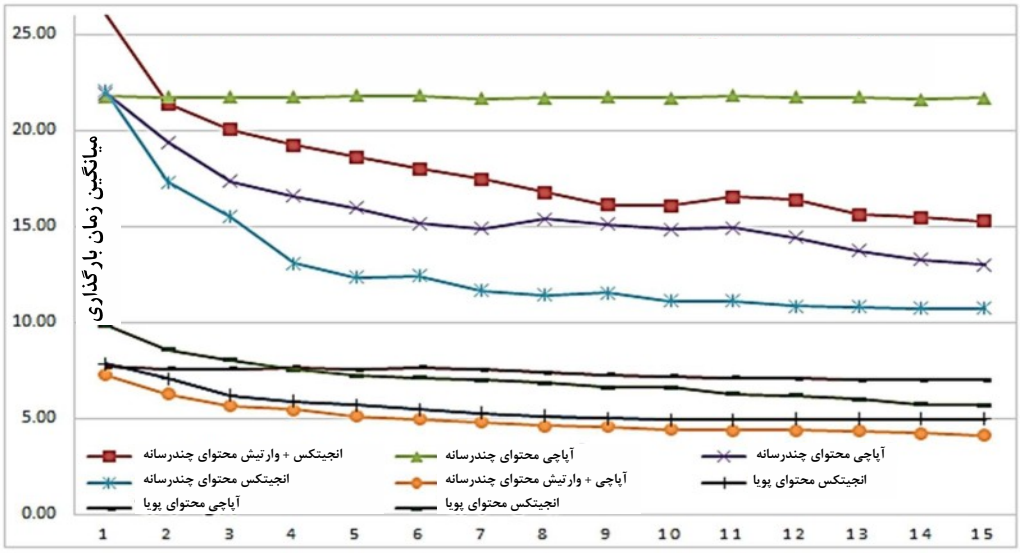
(نمودار-۸): آزمون زمان بارگذاری محتوای پویا با سرور انجینکس + وارنیش

(Chart-8): The test of time to load dynamic content with Nginx + Varnish server

به‌عنوان جمع‌بندی نهایی ابتدا میانگین دورهای آزمون در هر مورد آزمون را محاسبه کرده و همه نتایج را باهم در نمودار (۹) ملاحظه می‌کنیم. با مقایسهٔ نهایی عملکرد سامانه‌های وب‌سرور براساس محتوا، می‌توانیم ترتیب زیر را برحسب نوع محتوا و انواع مختلف سرورها ارائه کنیم:

محتوای ایستا چندرسانه‌ای: انجینکس نخست، آپاچی + وارنیش دوم، آپاچی سوم و انجینکس + وارنیش چهارم

محتوای پویا و پردازشی: آپاچی + وارنیش نخست، انجینکس دوم، انجینکس + وارنیش سوم و آپاچی چهارم



(نمودار-۹): میانگین زمان بارگذاری سرورهای مختلف با محتواهای پویا و ایستا

(Chart-9): The average of time to load different servers with dynamic and static contents

۵- نتیجه‌گیری

در این پژوهش چهار وب‌سرور آپاچی، آپاچی + وارنیش، انجینکس، انجینکس + وارنیش پیاده‌سازی و از نظر سرعت پاسخ آزموده شده‌اند. برای عملیات آزمون از سه سامانهٔ سنجش زمان مختلف کمک گرفتیم تا نتایج دقیق‌تر و به واقعیت نزدیک‌تر باشند. هر وب‌سروری با دو نوع محتوای پویا و ایستا بررسی شده است. همچنین هر وب‌سرور با محتوای پویا یا ایستا در شش دور آزمون شده و میانگین گرفته شده است تا اگر نتیجه‌ای به‌خاطر خطا در اینترنت، دور از واقعیت بود دفعات بعد جبران شود.

با توجه به جداول و نمودارهایی که حاصل شد، نتیجه می‌گیریم که برای محتوای ایستای چندرسانه‌ای و بدون پردازش سمت سرور، وب‌سرور انجینکس دارای بهترین عملکرد و پس از آن آپاچی + وارنیش است. در این مقایسه وب‌سرور آپاچی و وب‌سرور انجینکس + وارنیش دارای عملکرد ضعیف‌تری بودند؛ همچنین بر مورد محتوای پویا و پردازشی پرسرعت‌ترین وب‌سرور آپاچی + وارنیش بود و بعد از آن انجینکس قرار دارد.

برای پژوهش بیشتر پیشنهاد می‌شود، همین مسئله از زوایای دیگر از قبیل میزان حافظهٔ استفاده‌شده و میزان بار روی پردازندهٔ سمت سرور مورد پژوهش قرار گیرد؛ همچنین همین مسئله از منظر تعداد کاربران هم‌زمان در سرور و به‌علاوه کارایی سرورها از نظر فناوری شبکه توزیع محتوا نیز مورد بررسی قرار گیرد.

۶-مراجع 6-References

[1] M. Firtman, *High Performance Mobile Web: Best Practices for Optimizing Mobile Web Apps*. O'Reilly Media, 2016.

[2] J. Wagner, *Web Performance in Action: Building Faster Web Pages*. Manning Publications, 2017.

[3] D. Aivaliotis, *Mastering Nginx*. Packt Publishing Ltd, 2016.

[4] T. Feryn, *Getting Started with Varnish Cache: Accelerate Your Web Applications*, O'Reilly Media, 2017.

[5] Apache Foundation, *Apache HTTP Server 2.2 Official Documentation*. Vol. I-III, Apache Software Foundation, 2010.

[6] S. Corona, *Nginx: A Practical Guide to High Performance*, O'Reilly Media, 2017.

[7] R. Moutinho, *Instant Varnish Cache How-to*. Packt Publishing Ltd., New York, 2013.

[8] P. H. Kamp, You're doing it wrong. Communications of the ACM vol. 53 no.7, pp. 55-59, 2010.

[9] Web Page Test, Test a website's performance, Available: http://www.webpagetest.org/, [Accessed: Sept. 1, 2015].

[10] GTmetrix, Website Speed and Performance Optimization, Available: http://gtmetrix.com/, [Accessed: Sept. 1, 2015].

[11] Pingdom, Website & Performance Monitoring, Available: http://tools.pingdom.com/fpt/, [Accessed: Sept. 1, 2015].

[12] S. Bakhtiyari, Performance Evaluation of the Apache Traffic Server and Varnish Reverse Proxies, Master Thesis, Dept. of Informatics, Univ. of Oslo, Oslo, Norway, May 23, 2012.

[13] L. D. Tobias, Caching HTTP A comparative study of caching reverse proxies Varnish and Nginx, Master Thesis, School of Informatics, Univ. of Skövde, Skövde, Sweden, June 8, 2014.

[14] N. Mathieu, *Magento Site Performance Optimization,* Packt Publishing, 2014.

[15] B. T. Teklehaimanot, Virtualization of Video Streaming Functions. Ph.D. diss., Dept. of Computer Sciences, Univ. of Saarlandes, Saarbrücken, Germany, 2016.

[16] D. Kumar, V. Tyagi, and M. Vijay, With High Level Fragmentation of Dataset, Load Balancing in Cloud Computing. Int. Journal of Innovations & Advancement in Computer Science, Vol 3, Issue 5, 2014.

[17] R. DiCosmo, S. Zacchiroli and G. Zavattaro, Aeolus: A component model for the cloud. J. Information and Computation, Vol. 239, pp. 100-121, 2014.

[18] A. Grocevs, N. Prokofyeva, Modern approaches to reduce webpage load times, J. Env. Tech. Resources, 2015, Vol. 3, pp. 87-91, 2015.

[19] J. Andjarwirawan, I. Gunawan and E. Kusumo, Varnish Web Cache Application Evaluation. Proc. Intelligence in the Era of Big Data, 4th Int. Conf. on Soft Computing, Intelligent Systems, and Information Technology, pp. 404-410, Springer, 2015.

[20] A. Loechel and S. Schmid, Comparison of Different Caching Techniques for High-Performance Web Map Services. Int. Journal of Spatial Data Infrastructures Research, Vol. 8, pp. 43-73, 2013.

[21] J. Nogueira, D. Gonzalez, L. Guardalben and S. Sargento. Over-The-Top Catch-up TV content-aware caching. 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 1012-1017. IEEE, 2016.

[22] P. A. Baeza, Integración de un proxy inverso Varnish Cache con un panel de control Virtualmin para crear una plataforma de servicios de hospedaje Web. Thesis, Dept. Tècnica Sup-

erior d'Enginyeria Informàtica, Univ. Politècnica de València, Valencia, Spain, 2015.

[23] L. P. Petroski, J. Matos, and J. E. Bertotti. Analysis of an event oriented web server as a reverse proxy with flexible load distribution. Iberoamerican Journal of Applied Computing, Vol. 4, no. 2, 2016.

[24] A. Martínez-Álvarez, S. Cuenca-Asensi, A. Ortiz, J. Calvo-Zaragoza, and L. A. V. Tejuelo, Tuning compilations by multi-objective optimization: Application to apache web server, Journal of Applied Soft Computing, Vol. 29, pp. 461-470. 2016.

[25] D. S. Berger, K. R. Sitaraman and M. Harchol-Balter. AdaptSize: Orchestrating the Hot Object Memory Cache in a Content Delivery Network. 14th USENIX Symposium on Networked Systems Design and Implementation, pp. 483-498. 2017.

[26] R. Viscomi, A. Davies and M. Duran: *Using WebPageTest, Web Performance Testing for Novices and Power Users*. O'Reilly Media, October 2015.

[27] L. Čegan, Intelligent preloading of websites resources based on clustering web user sessions. 5th International Conference on IT Convergence and Security, pp. 1-4. IEEE, 2015.

[28] L. Čegan, Performance Evaluation of Preloading Resources for Web Pages, Advanced Computer and Communication Engineering Technology. Lecture Notes in Electrical Engineering, Vol. 362. Springer, 2016.

سنیه دیلمی مدرک کارشناسی خود را در رشتهٔ مهندسی رایانه از دانشگاه قم در سال ۱۳۹۰ دریافت کرد. او در سال ۱۳۹۱ وارد مقطع کارشناسی ارشد مهندسی فناوری اطلاعات گرایش تجارت الکترونیک دانشگاه قم شد. ایشان پایان نامهٔ خود را روی موضوع کارآمدی فناوری وب در اجرای سامانه‌های تجارت به مشتری و مشتری به مشتری در سال ۱۳۹۴ انجام دادند. زمینه‌های پژوهشی مورد علاقهٔ ایشان بهینه‌سازی و کارآمدی سامانه‌های تجارت الکترونیک در فناوری وب و سامانه‌های وب‌سرور است.

نشانی رایانامهٔ ایشان عبارت است از:

s.deylami1990@gmail.com

یعقوب فرجامی متولد ۱۳۴۸ هجری شمسی، مدرک کارشناسی، کارشناسی ارشد و دکترای تخصصی خود را در رشته ریاضی به‌ترتیب در سال‌های ۱۳۷۰، ۱۳۷۳ و ۱۳۷۷ دریافت کرد. وی استادیار و عضو هیئت علمی گروه مهندسی کامپیوتر و فناوری اطلاعات دانشگاه قم است. زمینه‌های پژوهشی مورد علاقهٔ ایشان مدل‌بندی سامانه‌های پیچیده، امنیت اطلاعات، هوش تجاری، اینترنت اشیا و هوشمندسازی سامانه‌ها است.

نشانی رایانامهٔ ایشان عبارت است از:

farjami@qom.ac.ir