راهکاری نوین برای جلوگیری از عدم سازگاری در به روز رسانی پویای سیستم‌های نرم‌افزاری مبتنی بر وب

سیدحبیب سیف زاده[۱] – حسن ابوالحسنی[۲] – محسن صدیقی مشکنانی[۳]

(۱) دانشجوی دکتری - دانشکده فنی مهندسی، گروه کامپیوتر، دانشگاه آزاد اسلامی واحد علوم و تحقیقات، تهران

(۲) دانشیار - دانشکده مهندسی کامپیوتر، گروه نرم افزار، دانشگاه صنعتی شریف

(۳) استادیار - دانشکده علوم و مهندسی، دانشگاه صنعتی شریف، شعبه پردیس بین الملل

تاریخ دریافت: تابستان ۱۳۹۱ تاریخ پذیرش: بهار ۱۳۹۲

خلاصه: به روز رسانی نرم‌افزار نیاز دارد تا برنامه‌ی در حال کار متوقف شده، تغییرات در آن اعمال گشته و سپس برنامه از ابتدا آغاز به کار نماید. این چرخه باعث بروز وقفه در اجرای نرم‌افزارها می‌شود که برای کاربران نهایی مطلوب نیست. این مساله خصوصا در مورد برنامه‌های وب که امروزه در اکثر صنایع مورد استفاده قرار می‌گیرند از اهمیت بیشتری برخوردار است، زیرا این برنامه‌ها اغلب باید به طور شبانه روزی در دسترس باشند. سیستمهای به روز رسان پویایی وجود دارند که امکان به روز رسانی نرم‌افزارها را در حین کار و بدون توقف آنها فراهم می‌آورند، لیکن چنین سیستم‌هایی برای به روز رسانی پویای نرم‌افزارهای وب نایاب هستند. در این مقاله می‌کوشیم راهکارهای لازم را برای تولید یک سیستم به روز رسان پویای مبتنی بر وب ارائه نماییم. در ارائه این راهکارها، ضمن روان‌سازی روش‌های موجود در ادبیات تحقیقِ به روز رسانی پویا برای استفاده در نرم‌افزارهای مبتنی بر وب، همواره تاکید بر حفظ سازگاری برنامه‌ها بوده است. برای این منظور، کلیه حالاتی که ممکن است در حین به روز رسانی پویای یک نرم‌افزار رخ دهد پیش‌بینی شده و اقداماتی که سیستم باید برای حفظ سازگاری خود و برنامه‌ی در حال اجرا در هر یک از حالات پیش‌بینی شده انجام دهد بیان گردیده است. در پایان نحوه پیاده‌سازی و ارزیابی سیستم پیشنهادی مورد بحث قرار گرفته است.

کلمات کلیدی: نگهداری نرم‌افزار، سیستم‌های به روز رسان پویا، قابل دسترس بودن.

۱- مقدمه

امروزه، به روز رسانی نرم‌افزار امری اجتناب‌ناپذیر است. این مسأله به خاطر رفع خطاهای احتمالی در برنامه‌ها یا وجود نیازهای جدید در سازمان‌ها صورت می‌گیرد. در به روز رسانی سنتی نرم افزار، نیاز داریم تا سیستم در حال اجرا را متوقف نموده، آن را با نسخه جدید جایگزین کرده و نسخه جدید را مجددا اجرا نماییم. این امر باعث قطع سرویس در زمان به روز رسانی می‌شود که برای کاربران نهایی ناخوشایند خواهد بود. این مشکل در سیستم‌های مبتنی بر وب مهم‌تر است زیرا قابل دسترس بودن برای این دسته از سیستم‌ها اهمیت بیشتری دارد. سیستم‌های وبی که در حال حاضر در تمامی صنایع، ادارات و سازمان‌ها مستقر هستند از چنین مشکلی رنج می‌برند.

سیستم‌هایی به نام «سیستم‌های به روز رسان پویا» وجود دارند که قادرند برنامه‌ها را در زمان اجرایشان و بدون توقف به روز نمایند [۱–۸]. این سیستم‌ها قادرند حالت نسخه قدیمی برنامه‌ها را در نسخه جدید آنها بارگذاری نمایند تا کاربران نهایی بدون متوجه شدن از به روز رسانی بتوانند به کار خود با سیستم ادامه دهند [۱]، [۳]، [۴]، [۶]، [۹–۱۱].

تا جایی که اطلاع داریم، مفهوم به روز رسانی پویا از سیستم اولیه‌ای که توسط فابری ارائه شد به وجود آمده و توسط بعضی از محققان دیگر ادمه پیدا کرد [۱۲]، [۱۳].

پس از آن، سیستم‌های به روز رسان پویا در حوزه‌های مختلف به وجود آمدند. از جمله آنها می‌توان به سیستم‌های به روز رسانی که می‌توانند سیستم‌های عامل [۵]، [۱۴–۱۷]، سیستم‌های بلادرنگ و ادغام شده

[۸], [۱۸–۲۱]، برنامه‌های رومیزی [۴], [۹], [۲۲–۲۵]، خدمتگزارها [۱], [۳], [۶], [۱۰], [۲۶–۲۹] و سیستم‌های توزیعی [۱۱], [۳۰–۳۲]، را به روز نمایند اشاره نمود. لیکن تاکنون هیچ کدام درحوزه سیستم‌های مبتنی بر وب فعالیتی نداشته‌اند [۷], [۳۳].

بررسی فوق از سیستم‌های به روز رسان پویای موجود نشان می‌دهد که سیستم به روز رسان پویایی که سیستم‌های نرم‌افزاری مبتنی بر وب را در زمان اجرا به روز کند تا کنون وجود ندارد. این در حالی است که به روز رسانی پویا برای سیستم‌های مبتنی بر وب از اهمیت بیشتری نسبت به برنامه‌هایی که تاکنون راهکارهای به روز رسانی پویا در مورد آنها ارائه شده است برخوردار است. دلیل آن این است که سیستم‌های مبتنی بر وب معمولا باید به صورت ۲۴ ساعته در حال کار باشند. احساس نیاز به یک سیستم به روز رسان پویا برای برنامه‌های مبتنی بر وب توسط محققان دیگر در این زمینه نیز بیان شده است [۷], [۳۳].

علاوه بر مساله فوق، سیستم‌های به روز رسان پویا عموما از مشکل عدم سازگاری در طول و یا بعد از به روز رسانی رنج می‌برند [۴], [۲۲], [۳۴], [۳۵]. این مشکل از آنجا ناشی می‌شود که ممکن است در طول مدت به روز رسانی، نسخه‌های مختلفی از یک مولفه در حافظه وجود داشته باشند و این باعث شود تا مولفه‌هایی که هنوز به روز نشده‌اند با مولفه‌هایی که عمل به روز رسانی آنها به اتمام رسیده است تعامل پیدا کرده و یک ناهماهنگی در اجرای برنامه به وجود آورند.

برای جلوگیری از مخاطرات احتمالی در طول مدت به روز رسانی، تعدادی از سیستم‌های به روز رسانی پویا عمل به روز رسانی را تا یک زمان از پیش تعیین شده به تعویق انداخته و هنگامی که آن زمان فرا رسید به صورتی تجزیه ناپذیر سیستم در حال اجرا را به روز می‌نمایند [۱], [۵], [۶], [۲۴], [۲۶], [۲۷], [۳۶]. اگر چه این کار باعث می‌شود تا مشکل ناسازگاری فوق‌الذکر اتفاق نیفتد، لیکن خود دو مشکل عمده دارد: اول اینکه به تعویق انداختن زمان به روز رسانی، سیستم‌ها را غیر قابل پیش بینی می‌کند، زیرا دقیقا معلوم نیست که چه زمانی شرایط لازم برای به روز رسانی فراهم شود، و دوم اینکه به دلیل اجرای تجزیه ناپذیر به روز رسانی‌ها، مدتی که سیستم در حال به روز رسانی است طولانی خواهد شد.

بنابراین، هدف اصلی این مقاله ارائه راهکاری برای به روز رسانی پویای سیستم‌های مبتنی بر وب با تاکید بر حفظ سازگاری و تا جای ممکن قابلیت پیش‌بینی بوده است.. برای رسیدن به این هدف، کلیه حالاتی که ممکن است برای یک سیستم به روز رسان پویای وب در طول حیاتش رخ دهد را بررسی کرده و برای هر حالت تمهیدات جلوگیری از مشکلات عدم سازگاری را اندیشیده‌ایم. همچنین در این مقاله، با استفاده از تکنیک‌های جلوکشیدن زمان به روز رسانی از یک سو و استفاده از تکنیک انتقال حالت با تاخیر [۳–۵], [۱۰] از سوی دیگر، سعی کرده‌ایم سیستم ارائه شده را قابل پیش‌بینی کرده و مدت به روز رسانی را در آن به حداقل برسانیم. چگونگی رسیدن به این اهداف در بخش بعدی بیان شده است. ذکر این نکته نیز مهم است سیستم‌های هدف این مقاله، برنامه‌های وب نوشته شده توسط زبان PHP بوده که فاقد پایگاه داده هستند.

ساختار بخش‌های پیش رو در این مقاله به صورت زیر است: در بخش (۲)، روش‌هایی که برای به روز رسانی پویای سیستم‌های مبتنی بر وب برگزیده‌ایم را بیان می‌کنیم، فصل (۳) نحوه پیاده سازی روش‌های مطرح شده در فصل (۲) و مشکلاتی که در این راه ممکن است روبرو شویم را مورد بحث قرار می‌دهد. فصل (۴) مباحثی در خصوص نحوه ارزیابی سیستم پیشنهادی با استفاده از آزمایشات تجربی که قصد انجام آنها را داریم مطرح می‌نماید. نهایتا، در فصل (۵) خلاصه‌ای از کارهای انجام شده در این مقاله و مواردی که نیاز به کار بیشتر دارند بیان می‌گردند.

۲– روش پیشنهادی

در این بخش، ایده‌های اصلی مطرح شده در سیستم پیشنهادی مطرح خواهند شد. در هر زیربخش ابتدا ایده‌های مطرح شده‌ی موجود در آن زمینه مرور شده و سپس بر اساس این ادبیات ایده مطرح شده در این مقاله توضیح داده شده و توجیه می‌گردد. در زیر بخش اول، مدلی که از طریق آن سیستم به روز رسان پویا صفحات تغییریافته را به روز می‌کند بیان می‌گردد. در زیربخش دوم، روشی که سیستم پیشنهادی استفاده می‌کند تا زمان مناسب به روز رسانی را بیابد و چگونگی همگرایی به این زمان بیان می‌شود. در زیربخش سوم مکانیزم انتقال حالتی که باعث می‌شود حالت برنامه وب و صفحات تغییر یافته بعد از به روز رسانی حفظ شود توضیح داده می‌شود. در پایان این بخش، محتویات وصله‌ای که باید توسط برنامه‌نویس فراهم شده و به سیستم به روز رسان پویا ارسال شود تا عمل به روز رسانی در میزبان وب اتفاق افتد توضیح داده می‌شود.

۲-۱- مدل به روز رسانی

در سیستم‌های به روز رسان پویایی که برای برنامه‌های رومیزی طراحی گردیده‌اند، دو مدل به روز رسانی از همه پرکاربردتر بوده‌اند [۳۳]: (۱) مدلی که با نام به روز رسانی تجمعی می‌شناسیم و عبارت از این است که در هنگام به روز رسانی کل قطعات برنامه فارغ از اینکه تغییر پیدا کرده‌اند یا خیر به روز شوند [۲۶], [۲۹] و (۲) مدل به روز رسانی افزایشی که در آن تنها قطعات تغییر پیدا کرده به روز خواهند شد [۱], [۳], [۵], [۲۳], [۲۵]. یکی از معمول‌ترین راههای توسعه سیستم‌های به روز رسان افزایشی، استفاده از دستیابی غیرمستقیم در برنامه‌هاست [۱], [۳], [۵], [۲۳], [۲۵]. برای این منظور، کلیه درخواست‌ها بین قطعاتِ برنامه از بخشی به نام به روز رسان پویا عبور می‌کند تا آن بخش درخواست را به نسخه مناسب از قطعه مقصد رسانده یا هر اقدام دیگری که مناسب می‌داند انجام دهد.

هرکدام از روش‌های فوق مزایا و معایبی دارد. بزرگترین مزیت روش اول این است که در طول اجرای برنامه هیچ کاهش کارآیی رخ

نمی‌دهد زیرا درخواست‌ها بدون هیچ واسطه‌ای از قطعه مبدا به قطعه مقصد حرکت می‌کنند، و عیب آن این است که مدت زمان به روز رسانی را طولانی‌تر می‌نماید، زیرا باعث جابجایی‌های غیرضروری هنگام به روز رسانی می‌گردد [۳۳]. این درحالی است که در روش دوم به دلیل اینکه هر درخواست باید از یک قطعه به روز رسان عبور کند کارآیی برنامه در زمان اجرا کاهش پیدا کرده ولی مدت به روز رسانی کوتاه‌تر است زیرا تنها قطعه‌هایی از برنامه که تغییر پیدا کرده‌اند جایگزین می‌شوند.

بکار بردن هر یک از روش‌های فوق در سیستم‌های وب مشکلاتی را پیش می‌اورد که عدم توجه به آنها و ارائه راهکارهای مناسب برای حل این مشکلات، استفاده از آنها را در سیستم‌های مبتنی بر وب با شکست مواجه می‌کند.

در مورد به روز رسانی‌های تجمعی، به دلیل اینکه در سیستم‌های وب ممکن است در هر لحظه از زمان تعدادی صفحه در حال ملاقات باشند، به روز رسانی کلیه قطعات برنامه به این معنا خواهد بود که صفحات در حال ملاقات نیز به روز گردند. از آنجا که کد صفحات در حال ملاقات در ماشین‌های مشتریان و دور از کد سایر قسمت‌های برنامه است، به روز رسانی تجمعی در سیستم‌های مبتنی بر وب بر خلاف مشابه‌های رومیزیشان کار دشوارتری است. البته این کار با اضافه کردن کدی مانند کدهای جاوا اسکریپت به کلیه صفحات برنامه به این منظور که مرتبا رسیدن به روز رسانی جدید را از خدمتگزار پرسیده و در صورتی که به روز رسانی جدیدی از راه رسیده بود آن را اعمال کنند امکان‌پذیر خواهد بود، لیکن این روش چند مشکل دیگر نیز دارد: اولا ترافیک بین خدمتگزار و ماشین مشتری را بالا می‌برد، دوما به دلیل اینکه باید کدی به کلیه صفحات برنامه الصاق شود دستکاری‌های زیادی باید در کد اصلیِ برنامه‌نویس انجام داد، و سوم اینکه به خاطر عدم وجود سیستمی متمرکز که تمام درخواست‌ها از آن عبور کند، نمی‌توان از تکنیک‌های جلوکشیدن زمان به روز رسانی مانند در صف قرار دادن درخواست‌ها در طول مدت به روز رسانی استفاده نمود.

روش دوم یا همان مدل دستیابی غیر مستقیم، در برنامه‌های رومیزی باعث می‌شود تا تمام دسترسی‌ها مانند فراخوانی توابع، دسترسی به متغیرها و ... نیاز به دستکاری برای اضافه کردن دستیابی غیر مستقیم داشته باشند. این در حالی است که در برنامه‌های وب با امکاناتی که میزبان‌های وب در اختیار می‌گذارند[1] می‌توان کلیه درخواست‌ها را به سمت صفحه مورد نظر مانند صفحه کنترل کننده به روز رسانی پویا تغییرجهت داد. از این رو برخلاف برنامه‌های رومیزی، دستکاری‌های کد به منظور اضافه کردن امکانات به روز رسانی پویا در روش دوم و در سیستم‌های مبتنی بر وب به مراتب از روش اول کمتر خواهد بود. این یک مزیت قابل ملاحظه برای این روش محسوب می‌شود. همچنین، استفاده از یک صفحه واحد برای کنترل اجرا و به روز رسانی برنامه به حفظ سازگاری در و بعد از به روز رسانی کمک کرده و می‌تواند امکاناتی مانند در صف قرار دادن درخواست‌ها به منظور جلوکشیدن زمان به روز رسانی را نیز به ارمغان آورد. البته، این روش خود مشکلاتی را به دنبال دارد که مهمترین آنها عبارتند از: تحمیل سربار اضافی در زمان اجرا به دلیل اضافه کردن یک سطح دستیابی غیرمستقیم به درخواست‌ها و وابستگی به تنظیمات میزبان وبی که برنامه بر روی آن قرار دارد به منظور تنظیم تغییر جهت درخواست‌ها. در این مقاله، از مشکل اول این روش به خاطر محاسنی که در حفظ سازگاری به دست می‌دهد چشم‌پوشی می‌کنیم. مشکل دوم نیز به دلیل امکانات خوبی که میزبان وب آپاچی برای تغییر مسیر درخواست در اختیار قرار می‌دهد برایمان مشکل بزرگی ایجاد نخواهد کرد.

با توجه به مباحث فوق، در این مقاله روش اضافه نمودن یک سطح دستیابی غیرمستقیم را به عنوان مدل به روز رسانی برگزیده‌ایم. برای این منظور، صفحه‌ای به نام dsu.php را به کلیه برنامه‌های وب اضافه خواهیم نمود و میزبان وب آپاچی را به نحوی تنظیم می‌کنیم که کلیه درخواست‌ها اعم از درخواست بارگذاری یک صفحه و یا درخواست به روز رسانی پویا به این صفحه ارسال شده و این صفحه بر اساس وضعیت‌های موجود اقدامات بعدی را در مورد درخواست وارده انجام دهد.

۲-۲- زمان به روز رسانی و همگرایی

همان طور که در بخش (۱) نیز دیده شد، تعیین زمان مناسب به روز رسانی در سیستم‌های به روز رسان پویا مساله دشواری است [۱], [۶], [۲۸], [۲۹], [۳۷], [۳۸]. تا به امروز، مدل‌های مختلفی برای تعیین زمان مناسب به روز رسانی ارائه شده است. برخی از سیستم‌ها به روز رسانی را بلافاصله اعمال می‌کنند: یا با استفاده از به روز رسانی توابع فعال [۱۶], [۲۸], [۲۹] یا با استفاده از ادامه اجرای توابع قدیمی و فراخوانی نسخه‌های جدید از بعد از به روز رسانی [۴], [۲۲], [۳۴]. روش‌های اول عموماً پیاده‌سازی را دشوار می‌سازند و بنابراین توسط سیستم‌های کمتری مورد استفاده قرار گرفته‌اند در حالیکه روش‌های دوم همواره نمی‌توانند سازگاری برنامه‌ها را حفظ نمایند [۳۳].

روش‌های دیگر، زمان به روز رسانی را تا برقراری یک شرط مشخص به تعویق می‌اندازند. این شرط می‌تواند توسط برنامه‌نویس تعیین گردد [۱], [۳], [۶], [۲۵]، زمانی که هیچ تابع به روز شده‌ای فعال نباشد فرا رسد [۵], [۱۷], [۲۴], [۲۶], [۲۷]، توسط یک مهلت تعیین شود [۳۹], [۴۰]، یا هنگامی رخ دهد که هیچ تابع تغییریافته‌ای در هیچ یک از تراکنش‌های فعال برنامه شرکت نداشته باشد [۴۱], [۴۲].

تحقیقات نشان می‌دهد که روش «مهلت» و «هیچ تابع به روز شده‌ای فعال نباشد» نمی‌توانند همواره سازگاری را برقرار کنند [۳۳]. دو روش دیگر یعنی «تعیین زمان به روز رسانی توسط برنامه نویس» و «هیچ تابع تغییریافته‌ای در تراکنش فعال شرکت نداشته باشد» بسیار به یکدیگر شبیه هستند خصوصاً اگر برنامه‌نویسان آخرین دستور حلقه بی‌پایان اصلی را به عنوان نقطه مناسب به روز رسانی انتخاب نمایند [۳۳]. تا این لحظه، این دو روش سازگارترین روش‌های شناخته شده برای تعیین زمان مناسب به روز رسانی در سیستم‌های به روز رسان

پویا بوده‌اند. لیکن، در این سیستم‌ها دقیقاً معلوم نیست چه زمانی اجرا به نقطه به روز رسانی خواهد رسید و این باعث عدم قابلیت پیش‌بینی در این سیستم‌ها می‌گردد [۳۳]. لازم به ذکر است در سیستم‌های چندنخی مانند میزبان‌های وب، رسیدن تمامی نخ‌ها به نقطه مناسب به روز رسانی قبل از اینکه به روز رسانی بتواند اعمال شود موجب مدت‌های توقف طولانی و بالا بردن میزان عدم قابلیت پیش‌بینی خواهد شد [۲۸]، [۴۳].

به دلیل تاکیدی که بر حفظ سازگاری در این مقاله داشته‌ایم، برای تعیین زمان به روز رسانی از روش «نقاط تعیین شده توسط برنامه‌نویس» استفاده کرده‌ایم، هرچند که با بهینه سازی این روش و ترکیب آن با روش‌های دیگر علاوه بر فراهم آوردن امکان استفاده آن در برنامه‌های وبی، تا حد امکان سعی در کاهش مشکل عدم قابلیت پیش‌بینی در این روش داشته‌ایم. اولین بهینه‌سازی مربوط به نحوه اعلام نقطه به روز رسانی است. برای این منظور پیشنهاد می‌گردد در برنامه‌های وب به جای آنکه نقطه یا نقاط مناسب به روز رسانی توسط برنامه‌نویس معرفی شود، بلاک‌هایی از کد که با به روز رسانی در انحصار متقابل هستند نشانه روند. از آنجا که یک برنامه وب معمولا از تعدادی صفحه مجزا که فاقد حلقه بی‌پایان اصلی هستند تشکیل شده است، استفاده از روش پیشنهادی تضمین خواهد نمود که عدم اجرای یک صفحه مانع از اعمال به روز رسانی نمی‌شود، زیرا چنین صفحه‌ای هرگز وارد ناحیه‌ای از کد که با به روز رسانی در تعارض است نخواهد شد. به علاوه، روش فوق‌الذکر این امکان را برای برنامه‌نویس فراهم می‌آورد تا بدون اضافه کردن کد خاص به صفحاتی که اجرای آنها با به روز رسانی پویا در تعارض نیست، اجرای همزمان این صفحات و اعمال به روز رسانی‌ها را مجاز اعلام کند. این در حالی است که در روش اولیه «تعیین زمان به روز رسانی توسط برنامه‌نویس» و در هنگام به روز رسانی هیچ کد دیگری مجاز به اجرا نبود. لازم به ذکر است همزمانی اجرا و به روز رسانی پویا به جلوکشیدن زمان به روز رسانی در میزبان‌های وب چندنخی کمک می‌کند زیرا لازم نیست تمام صفحات به نقطه مشخصی رسیده و متوقف گردند تا به روز رسانی آغاز گردد؛ البته به شرط آنکه برنامه‌نویس به درستی بلاک‌های متعارض با به روز رسانی را شناسایی و معرفی نماید تا سازگاری نیز حفظ شود.

دومین بهینه‌سازی که در روش اولیه «تعیین زمان به روز رسانی توسط برنامه‌نویس» شده است در صف قرار دادن درخواست‌هایی است که بعد از رسیدن درخواست به روز رسانی و تا قبل از اتمام اعمال آن به سیستم وارد می‌شوند. قبول نکردن درخواست‌های جدید از یک سو و اغلب اوقات کوتاه بودن مدت زمان اجرای دستورات در هر صفحه وب از سوی دیگر باعث می‌شود صفحات وبی که اکنون در حال اجرای کدهای در تعارض با به روز رسانی هستند بالاخره از این ناحیه خارج شده و به روز رسانی به سیستم اعمال گردد. برعکس، قبول درخواست جدید ممکن است اعمال به روز رسانی را دچار گرسنگی نماید، زیرا همواره ممکن است با خروج یک صفحه مذکور، صفحه‌ای که به تازگی اجرا را شروع کرده به این ناحیه وارد شده و این کار تا ابد ادامه پیدا کند. مزیت دیگر درصف قرار دادن درخواست‌ها این است که هیچ درخواستی که از سوی کاربر نهایی به سیستم ارسال می‌شود رد نشده و بالاخره پاسخ داده خواهد شد. این مساله، حذف دخالت کاربر نهایی از امر به روز رسانی را بیش از پیش تامین می‌نماید.

۲-۳- انتقال حالت

سیستم‌های به روز رسان پویا معمولا از مکانیزمی به نام انتقال حالت استفاده می‌کنند تا از طریق آن حالت برنامه قدیمی را در برنامه جدید تزریق نمایند [۱]، [۳]، [۴]، [۶]، [۱۰]، [۲۳]، [۲۹]. این کار باعث می‌شود تا کاربر نهایی بدون توجه به اعمال به روز رسانی، به کارهایی که قبلا با سیستم انجام می‌داده ادامه دهد.

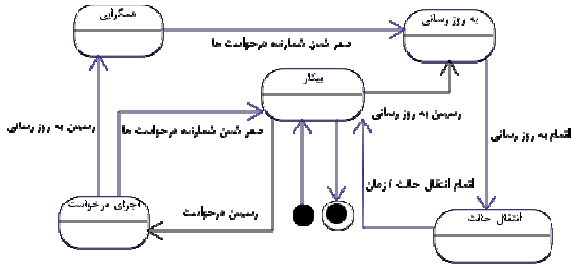
دو روش عمده انتقال حالت در ادبیات به روز رسانی پویا وجود دارند [۳۳]: (۱) روش مشتاق که در آن حالت کلیه مولفه‌های برنامه در مدت به روز رسانی انتقال پیدا می‌کند [۱]، [۹]، [۲۲]، [۲۶]، [۳۵] و (۲) روش با تاخیر که روش مدرن‌تری است و در آن، حالت هر مولفه بعد از به روز رسانی و در اولین دسترسی به آن انتقال پیدا می‌کند [۳–۵]، [۴۴]. هرچند استفاده از روش مشتاق در به روز رسانی پویا این مزیت را دارد که بعد از اعمال به روز رسانی هیچ سربار‌ی از نقطه نظر انتقال حالت به سیستم اعمال نشود، لیکن استفاده از این روش در سیستم‌های مبتنی بر وب دو مشکل اساسی دارد: (۱) به دلیل اینکه تعدادی از صفحات هم اکنون در حال ملاقات توسط کاربران نهایی هستند، روش مشتاق نیاز دارد تا حالت این صفحات نیز در زمان به روز رسانی انتقال پیدا کند و این به معنای مطلع شدن صفحات سمت مشتری از به روز رسانی‌های سمت خدمتگزار است. همان طور که قبلا نیز اشاره شد این کار نیازمند اضافه کردن کدهای اسکریپتی به تمام صفحات است که بنا به دلایلی که در در زیر بخش ۲-۱ ذکر شد قصد داریم از آنها جلوگیری نماییم. (۲) مشکل دوم این است که در این روش ممکن است انتقال‌های حالت غیر ضروری اتفاق افتد. رخ دادن این مشکل خصوصا در مورد برنامه‌های وب معمول‌تر است. به عنوان مثال، فرض کنید صفحه‌ای که اکنون در حال ملاقات است، پس از اتمام مطالعه توسط مرورگر بسته خواهد شد. انتقال حالت چنین صفحه‌ای غیر ضروری و مستلزم ترافیک بین کامپیوتر کاربر نهایی و خدمتگزار خواهد بود. در طرف مقابل، مشکلی که روش باتاخیر دارد این است که زمان دقیق اتمام انتقال حالت مشخص نیست. از آنجا که تا یک به روز رسانی و انتقال حالتش تمام نشود نمی‌توان به روز رسانی‌های بعدی را به سیستم اعمال نمود، روش انتقال حالت با تاخیر شاید منجر به رد درخواست‌های به روز رسانی بعدی گردد.

بر اساس مشکلات روش مشتاق که در بالا مشاهده شد، پیشنهاد می‌گردد که در به روز رسانی پویای سیستم‌های وب از انتقال حالت باتاخیر استفاده گردد هرچند که سعی خواهیم کرد از بروز مشکلات روش با تأخیر نیز جلوگیری کنیم. بر اساس روش پیشنهادی،

برنامه‌نویس کدهایی را به منظور انتقال حالت صفحات به روز شده به وصله‌ای که شامل صفحات تغییر یافته هستند الصاق کرده و آنها را در اختیار سیستم به روز رسان پویا قرار می‌دهد. این سیستم نیز پس از اعمال به روز رسانی در وضعیت انتقال حالت قرار گرفته و اگر درخواستی از صفحه در حال ملاقات قدیمی دریافت کند، پس از گذراندن درخواست از کدهای انتقال دهنده حالت، آن را به صفحه‌ی به روز شده تغییر جهت خواهد داد.

به منظور اینکه سیستم به روز رسان پویا تا ابد در وضعیت انتقال حالت نماند ترکیب دو راهکار پیشنهاد می‌شود: اول اینکه سیستم لیستی از صفحات در حال ملاقات نگه‌داری کند و پس از انتقال حالتِ هر صفحه قدیمی آن را از لیست حذف نماید. از آنجایی که تمام درخواست‌ها از سیستم به روز رسان پویا گذر می‌کند ذخیره‌سازی چنین لیستی کار پیچیده‌ای به نظر نمی‌رسد. بدین ترتیب، سیستم به روز رسان پویا تا زمانی که لیست فوق تهی نشده در حالت انتقال حالتِ بعد از به روز رسانی باقی می‌ماند. راهکار دوم، استفاده از مهلتِ متغیر جلسه در برنامه‌های وب است. همان طور که می‌دانیم چنانچه صفحه‌ای بیش از میزانی که مهلت متغیر جلسه تعیین می‌کند در حالت ملاقات بماند، درخواست‌هایش در صورتی که نیاز به ذخیره‌سازی یا بازیابی حالتی داشته باشند رد شده و کاربر آن صفحه بایستی کار با برنامه را از ابتدا آغاز کند. این بدان معناست که صفحه‌ای با این مشخصات دیگر نیازی به انتقال حالت توسط سیستم به روز رسان پویا ندارد. بنابراین در بدترین حالت، چنانچه صفحات در حال ملاقات دقیقاً قبل از رسیدن به روز رسانی درخواست شده باشند، به اندازه مدت متغیر جلسه فرصت دارند حالت خود را به روز کنند. در غیر این صورت، سیستم به روز رسان پویا مسئول انتقال حالت آنها نبوده و می‌تواند به روز رسانی‌های بعدی را پذیرش نماید. بدین ترتیب تضمین خواهد شد که حداکثر زمان عدم پذیرش به روز رسانی از سوی سیستمِ به روز رسان پویا به اندازه طول مدت جلسه تعریف شده در میزبان وب است.

با توجه به آنچه در این بخش دیده شد، نمودار حالت سیستم به روز رسان پویا مطابق شکل (۱) خواهد بود. همان طور که در این شکل دیده می‌شود سیستم به روز رسان پویا دارای ۵ حالت به نام‌های بیکار، اجرای درخواست، همگرایی، به روز رسانی، و انتقال حالت می‌باشد. حالت بیکار حالتی را نشان می‌دهد که سیستم هم اکنون به هیچ درخواستی پاسخ نمی‌دهد هرچند که ممکن است قبلاً به درخواست‌هایی پاسخ‌گفته باشد و بنابراین تعدادی صفحه‌ی قبلاً پردازش شده هم اکنون در حال ملافات در کامپیوترهای کاربران نهایی باشند. حالت اجرای درخواست حالتی را نشان می‌دهد که سیستم اکنون در حال اجرای دستوراتی که متعارض با به روز رسانی هستند می‌باشد. حالت همگرایی نشان‌دهنده حالتی است که سیستم در حال اجرای دستورات متعارض با به روز رسانی است در حالیکه درخواستی برای به روز رسانی برنامه نیز از راه رسیده است. در این حالت، سیستم درخواست‌های جدید را در صف قرار داده و منتظر می‌ماند تا اجرای کدهای متعارض با به روز رسانی به اتمام برسد. حالت به روز رسانی حالتی را نشان می‌دهد که سیستم در حال جایگزینی صفحات قدیمی با نسخه به روز شده‌شان و اجرای کدهای احتمالی است که باید در زمان به روز رسانی اجرا گردند. نهایتاً حالت انتقال حالت وضعیتی از سیستم را نشان می‌دهد که عمل جایگزینی صفحات وب با نسخه‌های به روز شده به اتمام رسیده ولی هنوز حالت تمام صفحات قدیمی به نسخه جدید انتقال پیدا نکرده است. شرایط لازم برای گذر از یک حالت به حالت دیگر در شکل موجود بوده و در این بخش نیز مورد بحث قرار گرفت؛ بنابراین از توضیح مجدد در رابطه با آنها خودداری می‌کنیم.



شکل (۱): نمودار حالت سیستم به روز رسان پیشنهادی

Fig 1: The Statechart diagram of the proposed dynamic updating system

Table (1): The response mechanism of the proposed updating system in different states

جدول (۱): نحوه پاسخ به درخواست در حالت‌های مختلف توسط سیستم پیشنهادی

نام حالت	درخواست صفحه	درخواست به روز رسانی
بیکار	بلافاصله انجام شود	بلافاصله انجام شود
اجرا	بلافاصله انجام شود	تا پایان انجام درخواست ها در حالت همگرایی بماند
همگرایی	در صف قرار داده شود	رد شود
به روز رسانی	در صف قرار داده شود	رد شود
انتقال حالت	درخواست جدید بلافاصله توسط برنامه به روز شده انجام شود	رد شود
	درخواست‌های در صف مانده توسط برنامه جدید انجام شوند	
	درخواست از صفحه قبلی پس از انتقال حالت توسط برنامه جدید انجام شود	

جدول (۱) نیز نشان می‌دهد که سیستم در مواجهه با درخواست به روز رسانی و یا درخواست اجرای یک صفحه در هر یک از حالات ۵ گانه فوق چه عملی را انجام خواهد داد. همانطور که در جدول مشاهده می‌شود، در هیچ یک از حالات، در خواست کاربر رد نمی‌شود بلکه ممکن است در بعضی از حالات در صف قرار داده شده و کمی با تأخیر

پاسخ داده شود. نکته دیگر اینکه تمامی درخواست‌های در صف مانده بلافاصله پس از جایگزینی صفحات قدیمی با نسخه جدید و ورود به وضعیت انتقال حالت توسط برنامه جدید اجرا شده و پاسخ داده خواهند شد.

۲-۴- مدل وصله

با توجه به روش پیشنهادی که در این بخش توضیح داده شد، برای به روز رسانی پویای سیستم‌های وب باید تغییراتی در کد اولیه برنامه اصلی به وجود آید. این تغییرات عبارتند از:

- تنظیم کردن میزبان وب (مانند آپاچی) به نحوی که کلیه درخواست‌ها را به سمت سیستم به روز رسان پویا (صفحه‌ای مانند dsu.php) هدایت کند، مگر آندسته از درخواست‌هایی که یک بار از این سیستم گذر کرده باشند.
- اضافه نمودن سیستم به روز رسان پویا (مانند dsu.php) به برنامه وب که شامل حالات ۵ گانه و عملکردی مانند آنچه در این بخش ذکر شد باشد.
- تعیین بلاک‌هایی از کد در صفحات وب که با به روز رسانی پویا در انحصار متقابل هستند.

با سه تنظیم فوق، یک برنامه وب قابلیت به روز رسانی پویا پیدا کرده و می‌تواند بر روی میزبان وب مستقر شود. پس از مستقرسازی و برای به روز رسانی پویای چنین برنامه‌ای، برنامه‌نویس باید بسته‌ای فراهم کند که شامل موارد زیر باشد:

- صفحات وب به روز شده با توجه به اینکه در آنها نیز بلاک‌هایی که با به روز رسانی پویا در تعارض هستند مشخص شده‌اند تا سیستم به روز رسان پویا را در تعیین زمان به روز رسانی‌های بعدی راهنمایی کنند.
- کدی شامل انتقال‌دهنده حالت متغیرهای سراسری برنامه مانند متغیر S_SERVER یا S_APPLICATION.
- فراهم آوردن کد انتقال دهنده حالت به ازای هر صفحه‌ی به روز شده به نحوی که سیستم به روز رسان پویا قادر باشد از طریق آنها حالت صفحه‌های قدیمی را به حالت جدید انتقال دهد.

موارد فوق توسط برنامه‌نویس و در قالب یک بسته تهیه شده و توسط درخواست HTTP به سیستم به روز رسان پویای مستقر در میزبان وب ارسال می‌شود. بقیه اعمال به روز رسانی بدون دخالت برنامه‌نویس و کاربر نهایی انجام خواهد گرفت.

۳- پیاده سازی و مشکلات پیش رو

پیاده‌سازی سیستم‌های به روز رسان پویا معمولا پیچیده و دارای مشکلات فراوانی است [۲۴]. در این بخش، سه مشکل از مهم‌ترین مشکلاتی که پیاده‌سازی راهکارهای ارائه شده در بخش ۲ می‌توانند ایجاد کنند را بیان نموده و برای هریک راهکارهایی ارائه می‌نماییم. در اولین زیربخش، پیاده‌سازی ایده‌ای که از طریق آن برنامه‌نویس بلاک‌های کدِ در انحصار متقابل با به روز رسانی پویا را معرفی می‌نماید ذکر می‌کنیم. در زیربخش دوم، اینکه چگونه سیستم به روز رسان پویا ممکن است باعث از دست رفتن پارامترهای درخواست در پروتکل HTTP شود و راه‌های مقابله با آن بیان می‌شوند؛ و در زیربخش آخر مشکل ایجاد حلقه بی‌پایان ناشی از تغییر مسیر درخواست‌ها به سمت سیستم به روز رسان پویا و راهکار مقابله با آن مطرح می‌گردند.

۳-۱- معرفی بلاک‌های متعارض با به روز رسانی

یکی از مسائلی که در روش پیشنهادی وجود دارد، نحوه بیان بلاک‌های متعارض با به روز رسانی پویا توسط برنامه‌نویس است. پیشنهاد می‌شود برای پیاده‌سازی این روش از متغیرهای سراسری که میزبان‌های وب در اختیار زبان‌های برنامه‌سازی وب پویا قرار می‌دهند استفاده گردد. به عنوان مثال، سیستم به روز رسان پویا شمارنده‌ای را در متغیر سراسری $_SERVER به صفر مقداردهی اولیه نماید. این شمارنده در اول هر بلاک متعارض با به روز رسانی پویا یک واحد افزایش و در پایان چنین بلاک‌هایی یک واحد کاهش می‌یابد. سیستم به روز رسان پویا تا صفر شدن مقدار چنین شمارنده‌ای اعمال به روز رسانی را به تعویق خواهد انداخت.

۳-۲- ارسال پارامتر

همانطور که می‌دانیم، درخواست‌ها در برنامه‌های وب ممکن است حاوی پارامترهایی باشد که از طریق دو متغیر POST و GET ارسال می‌شوند. تغییر جهت درخواست‌ها از مقصد اصلی به سیستم به روز رسان پویا باعث می‌شود تا تمامی پارامترهای درخواست به جای ارسال به مقصد درست به سیستم به روز رسان پویا ارسال شوند و در آنجا بدون استفاده از دست بروند. بنابراین، یکی از وظایف اساسی سیستم به روز رسان پویا بایستی این باشد که کلیه پارامترهای موجود در دو متغیر POST و GET را مجددا در درخواست خود به مقصد بعدی کپی کند. توجه شود که این کار باید به صورتی بازگشتی انجام شود زیرا دو متغیر فوق‌الذکر ممکن است حاوی آرایه‌های تو در تو از داده باشند. نویسندگان مقاله کدی با توضیحات فوق برای سیستم به روز رسان پویای در دست ساخت نوشته‌اند که برای تغییر مسیر درخواست‌ها در برنامه وب جوملا به درستی کار کرده است.

۳-۳- جلوگیری از بن بست

تغییر جهت کلیه درخواست‌ها به سمت صفحه به روز رسان پویا در خدمتگزار ممکن است باعث بروز یک حلقه بی‌پایان شود. به عنوان مثال، فرض کنید سیستم در حالت بیکار است و درخواستی برای اجرای یک صفحه مانند a.php به سیستم وارد می‌شود. به دلیل تنظیماتی که در میزبان وب وجود دارد این درخواست به سمت سیستم به روز رسان پویا مانند dsu.php تغییر جهت می‌دهد. این صفحه پس از اینکه متوجه می‌شود سیستم در حالت بیکار بوده و اجرای درخواست a.php بلامانع است درخواست را به سمت a.php

تغییر جهت می‌دهد. در این هنگام، میزبان وب درخواست a.php اخیر را به عنوان یک درخواست جدید تلقی کرده و مجددا آن را به سمت dsu.php تغییر جهت می‌دهد و این کار تا ابد ادامه پیدا خواهد کرد.
برای جلوگیری از مشکل فوق، صفحه به روز رسان پویا را به گونه‌ای پیاده‌سازی نموده‌ایم که پس از اقدامات لازم و هنگام تغییر جهت درخواست، عبارتی از پیش تعیین شده را به آدرس درخواست اولیه اضافه نماید. همچنین تنظیمات میزبان وب را به گونه‌ای انجام داده‌ایم که اگر عبارت فوق در آدرس درخواست بود، متوجه شود که این درخواست یک مرتبه از سیستم به روز رسان پویا عبور کرده و دیگر آن را به این سیستم ارسال ننماید. این مساله نیز بر روی سیستم‌های وب معروف مانند جوملا تست شده و مشکل خاصی در اجرای معمول برنامه نکرده است.

۴- ارزیابی

اولین موردی که می‌تواند در ارزیابی یک سیستم به روز رسان پویا به ذهن برسد نشان دادن کاربردی بودن چنین سیستمی است. تعدادی از اینگونه سیستمها کاربردی بودن خود را با به روز کردن برنامه‌های معمول و همه پسند مانند هسته سیستم عامل لینوکس در زمان اجرا به اثبات رسانده‌اند [۱۶]، [۱۷]. به علاوه، از آنجا که سیستم‌های به روز رسان پویا معمولا کدهایی را به منظور تسهیل در امر به روز رسانی به کد منبع برنامه‌ها اضافه می‌نمایند، میزان سرباری که این کدها به کارآیی برنامه اصلی تحمیل می‌کنند نیز همواره مورد توجه محققان بوده است. توسعه دهنده هر سیستم به روز رسان پویا بر اساس اینکه سیستمشان در چه حوزه‌ای عمل می‌کند محک‌های متفاوتی را برای اندازه‌گیری سربار سیستم خود انتخاب کرده‌اند. به عنوان مثال، سیستم‌های به روز رسان پویایی که خدمتگزارها را به روز می‌کنند معمولا خدمتگزارهای آپاچی، Vsftpd، Zebra و ... را در زمان اجرا به روز نموده و میزان سربار بر کارآیی این نرم‌افزارها را اندازه گرفته‌اند [۱]، [۳]، [۶]، [۱۰]، [۴۰]، سیستم‌هایی که در حوزه نرم‌افزارهای رومیزی فعالیت داشته‌اند نرم‌افزارهایی مانند کامپایلرها، محیط‌های برنامه‌سازی مجتمع و یا نرم‌افزارهای محاسبه‌گر مانند محاسبه سری فوریه را به انتخاب محک‌های خود برگزیده‌اند [۴]، [۲۴]، [۴۵]. در همین راستا، به تازگی مطالعه‌ای در زمینه معرفی و دسته‌بندی محک‌هایی که می‌تواند در حوزه به روز رسانی پویا مورد استفاده قرار گیرد انجام شده است [۴۶]. به هر حال، به دلیل اینکه تا به امروز به روز رسانی پویا در سیستم‌های وب کمتر مشاهده شده است، اکثر محک‌های مورد استفاده توسط محققان دیگر و یا معرفی شده توسط مطالعه مذکور برنامه‌های رومیزی یا خدمتگزار بوده و قابل استفاده برای ارزیابی سیستم پیشنهادی این مقاله نیستند. بر اساس این ادبیات، باید نرم‌افزارهایی جهت تست و ارزیابی سیستم‌های به روز رسان وب معرفی گردد. براساس تحقیقاتی که در این زمینه انجام داده‌ایم، یکی از محک‌های ممکن در این زمینه نرم‌افزار SquirrelMail است زیرا تماما با زبان PHP نوشته شده است، نرم‌افزاری شناخته شده است، کم حجم است، آرشیو چند سال از به روز رسانی‌های آن در اینترنت موجود است، تا به امروز در حال توسعه است و فاقد پایگاه داده‌هاست. از اینرو، قصد داریم پس از تکمیل فاز پیاده‌سازی، ایده‌های مطرح شده در مقاله را با نرم‌افزار SquirrelMail محک زده و میزان سربار این ایده‌ها را بر روی نرم‌افزار فوق‌الذکر اندازه‌گیری نماییم.

۵- نتیجه گیری و کارهای آینده

در این مقاله، راهکارهایی برای به روز رسانی پویای سیستم‌های نرم‌افزاری مبتنی بر وب ارائه گردید. تاجایی که می‌دانیم، به روز رسانی پویای این دسته از نرم‌افزارها برای اولین بار در ادبیات تحقیق در این مقاله مورد هدف قرار گرفته است. همچنین در این مقاله، راهکارهایی که قبلا در مورد به روز رسانی پویای سایر سیستم‌های نرم‌افزاری ارائه گردیده بود از دید سیستم‌های مبتنی بر وب مورد بحث قرار گرفت و برای اینگونه از سیستم‌ها روان‌سازی گردید. اگر مواردی از کارهای گذشته برای استفاده در برنامه‌های وب نیاز به بهینه‌سازی یا ترکیب با سایر روش‌ها داشت، این کار انجام شد. این مساله خصوصا در مورد مدل به روز رسانی و روش انتقال حالت پیشنهادی به خوبی مشهود است.
در ادامه، قصد داریم ضمن پیاده‌سازی کامل ایده‌های مطرح شده و ارزیابی آنها توسط نرم‌افزارهای وب موجود، سیستم پیشنهادی را در زمینه‌های زیر ارتقا دهیم:

- پشتیبانی از به روز رسانی پایگاه داده: از آنجا که تعداد زیادی از نرم‌افزارهای وب کنونی از سیستم پایگاه داده‌ها استفاده می‌کنند، عدم پشتیبانی از به روز رسانی پایگاه‌های داده در زمان اجرا باعث می‌شود سیستم به روز رسان پویا در به روز رسانی این قبیل از برنامه‌ها کاربرد نداشته یا با شکست مواجه شود. بنابراین، در آینده قصد داریم ضمن پشتیبانی از به روز رسانی پایگاه داده‌ها در زمان اجرا، سیستم پیشنهادی را در به روز رسانی برنامه‌های و بی مانند جوملا نیز مورد ارزیابی قرار دهیم.
- بهینه‌سازی امنیت و کارآیی: در بخش (۳) روشی برای جلوگیری از بن‌بست و ارسال پارامتر ارائه شد که طی آن سیستم به روز رسان پویا عبارتی مشخص را برای نشانه‌گذاری اینکه درخواست یک مرتبه از این سیستم عبور کرده است به آدرس اضافه نموده و تمام پارامترهای ارسالی به خود را برای سمت مقصد بعدی کپی می‌نماید. جابجایی پارامترهای درخواست بین صفحات برنامه وب از یک سو و دستکاری در آدرس درخواست‌های برنامه از سوی دیگر می‌تواند تهدیدی برای کارآیی و امنیت برنامه‌ها باشد. در آینده، قصد داریم سیستم پیشنهادی را در این دو مورد بهبود ببخشیم.

پی نوشت:

۱- مانند امکان Rewrite در میزبان وب آپاچی

References

[1] M. Hicks, S. Nettles, "Dynamic software updating", ACM Transactions on Programming Languages and Systems, Vol. 27, No. 6, pp. 1049–1096, Nov. 2005.

[2] D.Y. Lin, I. Neamtiu, "Collateral evolution of applications and databases", In IWPSE-Evol '09: Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, pp. 31–40, 2009.

[3] I.G. Neamtiu, "Practical dynamic software updating", University of Maryland, College Park, 2008.

[4] A.R. Gregersen, B.N. Jørgensen, "Dynamic update of Java applications - balancing change flexibility vs programming transparency", Journal of Software Maintenance and Evolution: Research and Practice, Vol. 21, No. 2, pp. 81–112, 2009.

[5] A. Baumann, "Dynamic update for operating systems", Computer Science & Engineering, Faculty of Engineering, UNSW, 2007.

[6] M. Hicks, K.S. Mckinley, "Dynamic software updates: A VM-centric approach", Update, No. 1.

[7] P. Bhattacharya, I. Neamtiu, "Dynamic updates for web and cloud applications", In APLWACA '10: Proceedings of the 2010 Workshop on Analysis and Programming Languages for Web Applications and Cloud Applications, pp. 21–25, 2010.

[8] M. Wahler, S. Richter, S. Kumar, M. Oriol, "Non-disruptive Large-scale Component Updates for Real-Time Controllers", In Proceedings of the 3rd International Workshop on Hot Topics in Software Upgrades, 2011.

[9] S. Malabarba, R. Pandey, J. Gragg, E. Barr, J.F. Barnes, "Runtime support for type-safe dynamic Java classes", Distributed Computing, 2000.

[10] H. Chen, J. Yu, R. Chen, B. Zang, P.C. Yew, "POLUS: A powerful live updating system", 29th International Conference on Software Engineering (ICSE'07), pp. 271–281, May 2007.

[11] Y. Vandewoude, "Dynamically updating component-oriented systems", 2007.

[12] R.S. Fabry, "How to design a system in which modules can be changed on the fly", In ICSE '76: Proceedings of the 2nd international conference on Software engineering, pp. 470–476, 1976.

[13] "Dymos: a dynamic modification system", 1983.

[14] C.A.N. Soules, J. Appavoo, K. Hui, R.W. Wisniewski, D.D. Silva, G.R. Ganger, O. Krieger, M. Stumm, M. Auslander, M. Ostrowski, B. Rosenburg, J. Xenidis, "System support for online reconfiguration", In USENIX 2003 Annual Technical Conference, pp. 141–154, 2003.

[15] S. Potter, J. Nieh, "AutoPod: Unscheduled system updates with zero data loss", In Second International Conference on Autonomic Computing, pp. 367–368, 2005.

[16] K. Makris, K.D. Ryu, "Dynamic and adaptive updates of non-quiescent subsystems in commodity operating system kernels", ACM SIGOPS Operating Systems Review, Vol. 41, No. 3, p. 327, Jun. 2007.

[17] J. Arnold, M.F. Kaashoek, "Ksplice: Automatic rebootless kernel updates", Design.

[18] J. Montgomery, "A model for updating real-time applications", Real-Time Systems, Vol. 27, No. 2, pp. 169–189, Jul. 2004.

[19] G. Gracioli, A.A. Fröhlich, "An operating system infrastructure for remote code update in deeply embedded systems", In Proceedings of the 1st International Workshop on Hot Topics in Software Upgrades, pp. 31–35, 2008.

[20] H. Seifzadeh, A.A.P. Kazem, M. Kargahi, A. Movaghar, "A method for dynamic software updating in real-time systems", In ICIS '09: Proceedings of the 2009 Eigth IEEE/ACIS International Conference on Computer and Information Science, pp. 34–38, 2009.

[21] A.C. Noubissi, J. Iguchi-Cartigny, J.L. Lanet, "Hot updates for Java based smart cards", In Proceedings of the 3rd International Workshop on Hot Topics in Software Upgrades, 2011.

[22] M. Dmitriev, "Towards flexible and safe technology for runtime evolution of java language applications", In In Proceedings of the Workshop on Engineering Complex Object-Oriented Systems for Evolution, in association with OOPSLA 2001 International Conference, 2001.

[23] A. Orso, A. Rao, M.J. Harrold, "A technique for dynamic updating of Java software", International Conference on Software Maintenance, Proceedings., pp. 649–658, 2002.

[24] R.P. Bialek, "Dynamic updates of existing Java applications", Interface, 2006.

[25] G. Bierman, M. Parkinson, J. Noble, "UpgradeJ: Incremental typechecking for class upgrades", pp. 1–25.

[26] D. Gupta, P. Jalote, "On-line software version change using state transfer between processes", Software: Practice and Experience, Vol. 23, No. 9, pp. 949–964, Sep. 1993.

[27] G. Altekar, I. Bagrak, P. Burstein, A. Schultz, "OPUS: online patches and updates for security", In SSYM'05: Proceedings of the 14th conference on USENIX Security Symposium, p. 19, 2005.

[28] K. Makris, R.A. Bazzi, "Immediate Multi-threaded dynamic software updates using stack reconstruction", In Proceedings of 2009 USENIX Annual Technical Conference, 2009.

[29] C.M. Hayden, E.K. Smith, M. Hicks, J.S. Foster, "State transfer for clear and efficient runtime upgrades", In Proceedings of the 3rd International Workshop on Hot Topics in Software Upgrades, 2011.

[30] S. Ajmani, B. Liskov, L. Shrira, "Modular software upgrades for distributed systems", In LNCS ECOOP 2006: Object-Oriented Programming, Vol. 4067, pp. 452–476, 2006.

[31] S. van der Burg, E. Dolstra, M.de Jonge, "Atomic upgrading of distributed systems", In HotSWUp '08: Proceedings of the 1st International Workshop on Hot Topics in Software Upgrades, pp. 1–5, 2008.
[32] V.P. La Manna, "Dynamic software update for component-based distributed systems", In Proceedings of the 16th international workshop on Component-oriented programming, pp. 1–8, 2011.
[33] H. Seifzadeh, H. Abolhassani, M.S. Moshkenani, "A survey of dynamic software updating", Journal of Software: Evolution and Process, Vol. 25, No. 5, pp. 535–568, 2012.
[34] G. Hjálmtýsson, R. Gray, "Dynamic C++ classes: A lightweight mechanism to update code in a running program", In ATEC '98: Proceedings of the annual conference on USENIX Annual Technical Conference, p. 6, 1998.
[35] H. Seifzadeh, M. Kermani, M. Sadighi, "Dynamic maintenance of software systems at runtime", Third International Conference on Availability, Reliability and Security, pp. 859–865, Mar. 2008.
[36] M. Jalili, "A hybrid model of dynamic software updating in C", Islamic Azad University of Shabestar, 2009.
[37] G. Bierman, M. Hicks, P. Sewell, "Formalizing dynamic software updating", Update, pp. 1–17, 2003.
[38] D. Gupta, P. Jalote, G. Barua, "A formal framework for on-line software version change", IEEE Trans. Softw. Eng., Vol. 22, No. 2, pp. 120–131, 1996.
[39] M. Wahler, S. Richter, M. Oriol, "Dynamic software updates for real-time systems", In HotSWUp '09: Proceedings of the Second International Workshop on Hot Topics in Software Upgrades, pp. 1–6, 2009.
[40] J. Stanek, S. Kothari, T. Nguyen, C. Cruz-Neira, "Online software maintenance for mission-critical systems", 22nd IEEE International Conference on Software Maintenance, pp. 93–103, Sep. 2006.
[41] K. Hui, C.A.N. Soules, R.W. Wisniewski, O. Krieger, M.A. Auslander, D.J. Edelsohn, B. Gamsa, G.R. Ganger, P. Mckenney, M. Ostrowski, B. Rosenburg, M. Stumm, J. Xenidis, "Enabling autonomic behavior in systems software with hot swapping", Vol. 42, No. 1, 2003.
[42] Y. Vandewoude, P. Ebraert, Y. Berbers, T. D'Hondt, "Tranquility: A low disruptive alternative to quiescence for ensuring safe dynamic updates", IEEE Trans. Softw. Eng., Vol. 33, No. 12, pp. 856–868, 2007.
[43] I. Neamtiu, M. Hicks, "Safe and timely updates to multi-threaded programs", SIGPLAN Not., Vol. 44, No. 6, pp. 13–24, 2009.
[44] C. Boyapati, B. Liskov, L. Shrira, C.H. Moh, S. Richman, "Lazy modular upgrades in persistent object stores", SIGPLAN Not., Vol. 38, No. 11, pp. 403–417, 2003.
[45] K. Makris, "Whole-program dynamic software updating", ARIZONA STATE UNIVERSITY, 2009.
[46] E.K. Smith, M. Hicks, J.S. Foster, "Towards standardized benchmarks for dynamic software updating systems", In Proceedings of the 4th International Workshop on Hot Topics in Software Upgrades, 2012.