الگویی نو در معماری نرم افزار مبتنی بر رویه های ذخیره شده

مهدی هوشمندی*[1]، دکتر بهروز معصومی [2]
کارمند واحد فناوری اطلاعات دانشگاه آزاد اسلامی قزوین ، iranerpgroup@gmail.com
هئیت علمی دانشگاه آزاد اسلامی قزوین ، Masoumi@Qiau.ac.ir

چکیده- الگوهای معماری متعددی معرفی شده است که باعث پیشبرد توسعه نرم افزار شده است، اما هر یک در کنار مزایا، معایبی داشته اند که باعث اقبال کمتر یا بیشتر به آنها شده است. الگوی مطرح شده به نام SP-Based اکثر مزایای الگوهای موجود را داشته و حداقل معایب را دارد و در اکثر محیط ها و کسب و کارها قابل استفاده می باشد. مهمترین مزیت این معماری کاهش چشمگیر خطوط برنامه نویسی در لایه های مختلف پیاده سازی سامانه های نرم افزاری میباشد. در ارزیابی صورت گرفته روی معماری MVC .net core بالای 80درصد حجم کدنویسی، تعداد فایل ها و از همه مهمتر هزینه اجرای پروژه کاهش پیدا کرده است

کلید واژه- الگوی معماری نرم افزار، رویه های ذخیره شده، منطق کسب و کار، معماری سه لایه، افزایش بهره وری

1- مقدمه

در چرخه حیات توسعه نرم افزار systems development life cycle (SDLC) **اساسی ترین فاز معماری نرم افزار است. همانند اسکلت یک ساختمان که به سازندگان کمک می کند بدرستی ساختمان را بنا نهند.**[1] **انتخاب یک الگو معماری نرم افزار مناسب جهت توسعه و ایجاد یک پروژه نقش کلیدی در با صرفه بودن بقاء آن دارد. بدون داشتن یک معماری شفاف و خوش تعریف بسیاری از توسعه دهندگان و معماران برای ادامه پروژه دچار مشکل خواهند شد. با انتخاب معماری هزینه و زمان اجرای پروژه قابل تخمین خواهد شد و برنامه ریزی مناسب برای ادامه پروژه انجام خواهد گرفت. معماری مانند چهارچوب و اسکلت ساختمان، بعد از طراحی و پیاده سازی به جزء تخریب قابل اصلاح نیست.**

آرمان معماران نرم افزار انتخاب معماری است که با کمترین هزینه و بالاترین سرعت محصولی با کیفیت ارائه دهند. الگو معماری SP-Based بی شک یکی از استثنائات دنیای معماری نرم افزار است که تاحدود زیادی به این آرمان نزدیک شده است و بسادگی با اکثر زبانها و بانک ها قابل پیاده سازی است.

در ادامه، مقاله چنین سازمان دهی شده است. بخش دوم کارهای پیشین در زمینه معماری نرم افزار بررسی شده است. بخش سوم روش پیشنهادی و اجزاء تشکیل دهنده و نحوه تعامل آنها شرح داده شده است. بخش چهارم بهبودهایی که این الگو معماری ایجاد می کند بررسی می شود. بخش پنجم ارزیابی و نتایج که در یک تحقیق میدانی حاصل شده است ارائه می شود. در بخش ششم نتیجه گیری و بخش هفتم مراجع استفاده شده معرفی می شود.

2- کارهای پیشین و بهبودهایی که حاصل کرده

برای درک بهتر شرایط موجود در حوزه معماری نرم افزار، چالش ها و فرصت ها، انواع الگوهای معماری مطرح جهت پیاده سازی پروژه های نرم افزاری در این بخش بررسی و با معماری مطرح بررسی شده است.

2-1- معماری سه لایه

متداول ترین نوع الگو معماری می باشد. ساختار سه لایه Model-View-Controller یا MVC یک روش توسعه استاندارد جهت توسعه در فریمورک های مطرح وب می باشد. در این معماری درخواست اجرای یک عملیات از لایه های مختلف عبور کرده تا به بانک اطلاعاتی می رسد. در این روش لایه مدل بالای بانک اطلاعاتی می باشد و شامل اطلاعاتی در رابطه با انواع داده ای بانک اطلاعاتی و منطق کسب و کار می باشد. در بالاترین سطح در این معماری، لایه View قرار دارد که اغلب شامل CSS، Java Script، HTMLو داده های داینامیکی می باشد که در زمان اجرا در View اضافه می شود. در وسط هم لایه کنترلر با انواع قواعد و روش ها جهت انتقال داده ها و درخواست ها از لایه View به لایه مدل وجود دارد

مقایسه با SP-Based: جداسازی و تفکیک کدها را داشته به همان ترتیب داشته، اما کندی موجود در توسعه را بخاطر رویکرد جنبه گرایی حل کرده است.

2-2- معماری رویدادگرا

در این معماری به تولید، شناسایی و واکنش به رویدادها اهمیت داده می شود و مکانیزمی را ایجاد می کند که رویدادها در سریعترین زمان پاسخ داده شوند. رویداد یک تغییر قابل شناسایی در وضعیت جاری سامانه است.

مقایسه با معماری SP-Based: مانند معماری رویدادگرا عملیات را با کمترین واسطه و به سرعت اجرا نموده، اما کنترل مرکزی موردنیاز در این معماری را ایجاد کرده و مزایای بسیاری به این واسطه به ارمغان آورده است.

2-3- معماری میکروکرنل

این معماری با داشتن یک هسته مرکزی کوچک حداقل ابعاد را داشته و plug-in محور است. در این معماری ویژگی ها و امکانات جدید از طریق نصب plug-in اضافه می شود.

مقایسه با معماری SP-Based: مانند این معماری دارای هسته کوچک و سبکی است، اما رویه های ذخیره شده ای که به عنوان plug-in یا سرویس به آن متصل می وشوند در قبال تغییرات هسته محدودیت ندارند.

2-4- معماری ابری

فاکتور محدود کننده در برنامه هایی با مقیاس بزرگ، بانک اطلاعاتی میباشد که جه میزان به درخواست های همزمان می تواند پاسخ دهد. این معماری توانسته بر این مشکل با استفاده از فضای حافظه اشتراکی فائق آید.

مقایسه با معماری SP-Based: به خاطر مکانیزم سوئچ پویا بین سرورها در خلال اجرای پروژه امکان پشتیبانی از کاربران بالا را داشته، اما برخلاف معماری ابری امکان استفاده داشتن داده پایداری کامل نیز دارد.

3- روش پیشنهادی

با توجه به معرفی معماری های مختلف و شناخت نقاط ضعف و قوت آنها، روش پیشنهادی برای بهبود روش های موجود، معماری SP-Based می باشد، در ادامه شرح کاملی از این معماری و اجزاء خواهیم داشت.

3-1- معماری SP-Based

معماری SP-Based مبتنی بر رویه های ذخیره شده و معماری جنبه گرایی عمل می کند. کدنویسی را به شدت کاهش می دهد و رویکرد ساده و موثری در کنترل سطح دسترسی و اجرای عملیات درخواستی دارد. شرایط تست را گذرانده در پروژه های بزرگی در حال استفاده می باشد.

3-2- اجزای تشکیل دهنده معماری SP-Based

اجزاء این معماری در سه بخش کلاینت، وب سرور و بانک تقسیم شده، که مسئولیت ارسال درخواست، پردازش و ارسال پاسخ را دارند. این اجزاء به چهار جزء زیر تقسیم می شوند.

3-2-1- موتور پردازشی رویداد

در سمت کلاینت، اعتبارسنجی و ساخت پارامترهای ورودی رویداد و نیز پردازش نتیجه اجرای رویداد در سرور را انجام میدهد. اجرای تمام عملیات در سرور در قالب

رویدادهایی است که در کلاینت رخ داده و از موتور پردازشی رویداد EPE (Event Processing Engine) گذشته است.

3-2-2- موتور اعمال سیاستها

موتور اعمال سیاستها PE (Policy Engine)، خط مقدم برخورد با درخواست های کلاینت بوده و طبق قواعد تعریف شده درخواست را اجرا می کند. ممکن است در همین مرحله درخواست برگشت برخورد و از ادامه اجرای آن جلوگیری به عمل آید. PE امکان مانتیور کردن وضعیت سامانه، اعمال قواعد مدنظر و ... را انجام می دهد. به عنوان مثال می تواند دسترسی به اجرای یک عملیات خاص را برای فرد و یا گروهی در مدت نامحدود یا محدود باز یا ببندد. عملیات را می تواند دسته بندی نماید و برخوردهای مختلفی روی آنها صورت دهد. به عنوان مثال می تواند برای عملیات هایی لاگ کردن را منتقی کند، برخی را بدون سطح دسترسی اجرا کند، برخی را در صورت بروز خطا با SMS یا notification به مدیرسامانه اطلاع دهد.

3-2-3- موتور اجرایی SP

بعد از عبور درخواست از لایه کنترلی یا PE، نوبت به اجرای عملیات می رسد. بدین منظور باید SP درخواستی اجرا شود. برای اجرا SP نیاز است که پارامترهای ورودی، پارامترهای خروجی، جزئیات SP و مقدار پارامترهای ورودی تعیین گردد، کلیه این عملیات توسط موتور اجرایی SP انجام شده و در نهایت خروجی مربوطه به کلاینت برگردانده می شود..

3-2-4-رویه های ذخیره شده

رویه های ذخیره شده یا SP (Stored Procedure) یک جزء استاندارد بانک های رابطه ای هستند که امکان اجرای دسته ای دستورات SQL را می دهند. SPها در معماری SP-Based جزء اساسی آن محسوب شده و منطق کسب و کار را در خود جای میدهند. استفاده از SP مزایای زیر را در معماری به دنبال داشته است:

- اجرای مستقیم دستورات SQL در هسته بانک و کاهش ترافیک شبکه
- کپسوله کردن منطق کسب و کار
- کشف آسان خطاهای احتمالی کدنویسی SQL
- امکان اعمال سطوح دسترسی مختلف برای SPها
- جلوگیری از حملات SQL-injection

4- بهبودهایی که حاصل کرده

معماری SP-Based از مزایای معماری پیشین استفاده و معایب آنها را تا حدود زیادی رفع نموده است. که در بخش سوم کارهای پیشین به آن اشاره شد. علاوه بر بهبودهای گفته شده مزایایی دیگری نیز این معماری به شرح زیر ایجاد نموده است.

4-1- حداقل کدنویسی در لایه های مختلف

معماری SP-Based با نگرش حنبه گرایی موفق شده است در معماری سه لایه معمول بسته به زبان انتخابی بالای 80٪ حجم کد نویسی و تعدا فایلها را کاهش داده و تاثیر مستقیمی روی هزینه و سرعت اجرای پروژه می گذارد. کنترل سطح دسترسی ساده تر شده و توسعه و رفع خطاهای ماژول ها و عملیات های مختلف ساده تر انجام می شود. علاوه تعداد فایلها پروژه باعث سهولت در تهیه نسخه پشتیبان و اشتراک گذاری پروژه شده است.

4-2- کاهش هزینه و وابستگی به نیروی متخصص

برای بقاء پروژه بایستی وابستگی به نیروی انسانی متخصص کاهش پیدا کند. چهارچوب و معماری طراحی شده برای پروژه نقش اصلی را در این بین بازی می کند. معماری SP-Based در این موضوع بسیار پیشرو بوده و بر عکس مدل معمول MVC که تولید نهایی یک فرم منوط به تولید مدل آن و زیر ساخت های لازم در بانک می باشد، در معماری کنونی این وابستگی به شکل چشم گیری بخاطر استفاده از ماهیت جنبه گرایی کاهش پیدا کرده و تغییر در نیروها به راحتی و کمترین آموزش امکان پذیر است. چون از زبان پایه ای SQL در لایه بانک، کدنویسی

اندک در لایه کنترلر و زبان های پایه ای HTML، CSS، جاوا اسکریپت و JQuery در لایه View استفاده می شود.

زمان انجام پروژه و نیروی انسانی موردنیاز نیز در این معماری با توجه به کاهش پیچیدیگی و تعداد خطوط پروژه بسیار کاهش یافته و طبق بررسی میدانی بسته به نوع معماری بین 50٪ تا 80٪ این معماری صرفه جویی ایجاد می نماید.

4-3- اجرای بهینه درخواست ها

در معماری های معمول برای اجرای درخواست با توجه به حجم بالای کد، پیچیدگی و لایه های متعدد آنقدر مسیر های مختلف و طولانی برای اجرای درخواست تعریف شده است که اعمال سیاست های توسعه ای با دشواری صورت می گیرد.

در معماری SP-Based این مسیر به شکل چشمگیری کوتاه و ساده شده است. این سادگی برعکس معماری های معمول امکان اعمال سیاست های توسعه ای به راحتی ممکن می کند. از جمله این سیاست ها می توان به کنترل سطح دسترسی، کنترل خطا، ثبت تراکنش های موفق و ... اشاره نمود.

4-4- کاهش شدید ترافیک شبکه

به خاطر ماهیت معماری SP-Based و عملکرد مبتنی بر رویه های ذخیره شده بنابراین بار ترافیکی بین سرورهای وب و بانک بسیار کاهش پیدا کرده و عملیات درخواستی کلاینت سریعا اجرا می شود. از طرفی فرمت انتقال دیتا باعث حداقل شدن طول دیتا تبادل شده با کلاینت می شود. این مزیت زمانی بیشتر اهمیت پیدا می کند که بخواهیم در قالب وب سرویس از خدمات سرورهای دیگر طبق معماری SP-Based استفاده کنیم.

4-5- بهره وری بالا در توسعه منطق کسب و کار

از آنجا که بیشتر منطق کسب و کار (Business Logic) در SP-Based در رویه های ذخیره شده قرار دارد، در نتیجه توسعه و کد نویسی در این بخش را فوق العاده سریع، ساده و ایمن نموده است. این معماری با امکان تقسیم منطق کسب و کار به کوچکترین اجزاء عملیاتی امکان اعمال سیاست های کنترلی را در دو سطح کاربران سامانه و توسعه دهندگان آن براحتی فراهم نموده است. اعمال سیاست های کنترلی در سطح توسعه دهندگان باعث مزایای فراوانی از جمله امکان تعریف زیرسیستم های مختلف و دادن مجوز های دسترسی توسعه دهندگان به بخشی از منطق کسب وکار شده است. بدین ترتیب امنیت و پایداری کدها افزایش بسزایی یافته است و از تغییرات غیر برنامه ریزی شده منطق کسب و کار جلوگیری به عمل می آید.

5- ارزیابی و نتایج

طبق یک بررسی میدانی چندین ماهه صورت گرفته روی یک پروژه با مقیاس بزرگ طبق فریمورک MVC .net core با دو معماری معمول و معماری SP-Based با پارامترهای مختلف نتایج جالبی کسب شد که در ادامه به آن خواهیم پرداخت.

مهمترین تفاوت این دو روش در نحوه دسترسی به داده هاست که در معماری معمول به صورت غیر مستقیم از طریق ORM اقدام می شود ولی در SP-Based به صورت مستقیم از کنترلر به رویه های ذخیره شده انجام می شود.

5-1- میزان کدنویسی

خطوط کدنویسی یا LOC (Line of Code) یک معیار مناسب برای محاسبه انعطاف و میزان بهره وری در یک پروژه می باشد. بدین منظور بخش های مختلف پروژه شناسایی شده و بر اساس تعداد عملیات و ماژول ها تعداد خطوط تخمین زده می شود. فرمول ها در هر یک از معماری ها با توجه به ارزیابی صورت گرفته تولید شده اند و تا حدود زیادی دقیق می باشند. سطح توانایی برنامه نویسان در هر دو گروه حرفه ای در نظر گرفته شده بنابراین تعداد خطوط تولیدی آنها با توجه به معماری مورد استفاده در حد حداقل می باشد. جدول شماره 1 به معرفی و مقداردهی پارامترهای موجود می پردازد. جدول شماره 2 و 3 به مقایسه تعداد خطوط این دو معماری در لایه های

مختلف می‌پردازد. درنهایت معماری مطرح شده باعث کاهش 88درصدی کدنویسی شده است.

پارامتر	MVC	SP-Based
M (تعداد کل ماژول ها)	300	300
O (تعداد عملیات حدود 5.3برابر ماژولها)	1600	1600
HTML (تعداد خطوط ماژول)	80	15
JS (تعداد خطوط ماژول و عملیات)	35	3
CSS (تعداد خطوط ماژول)	10	0
CL (خطوط لایه کنترلر عملیات)	7	0
BL (خطوط لایه کسب و کار عملیات)	4	0
DAL (خطوط لایه دسترسی به داده عملیات)	7	0
SP (تعداد خطوط متوسط هر SP)	20	3
ضریب	0.5	0.1

جدول 1: تعریف و مقدار پارامترهای های محاسبه تعداد خطوط و فایل

نام لایه	محاسبه خطوط MVC	تعداد
View	m(JS+CSS+HTML)	37500
کنترلر	o(CL)	11200
DAL و BL	o(BL + DAL)	17600
بانک	o(SP)(0.5)	16000
تعداد کل خطوط معماری		82300

جدول 2: محاسبه تعداد خطوط در معماری MVC .net core

نام لایه	محاسبه خطوط SP-Based	تعداد	کاهش
View	m(HTML)+o(JS)(0.1)	4980	٪86
کنترلر	ثابت	200	٪98
DAL و BL	ثابت	0	100 ٪
بانک	o(SP)	4800	٪70
تعداد کل خطوط معماری		9980	٪88

جدول 3: محاسبه تعداد خطوط در معماری SP-Based

5-2- تعداد فایلها

مدیریت فایل ها در پروژه بزرگ کار بسیار حساس و مهمی است. بطوری که اگر حتی تغییر کوچکی در فایلها بدرستی در سرور اعمال نشود، باعث ایجاد بعضاً اختلالات گسترده ای در برنامه می شود. برای این منظور از ابزاری مانند github استفاده می شود که با وجود تسهیل کردن مدیریت فایل، سربارهایی هزینه های و زمانی و آموزشی در پی دارد. با معماری SP-Based چون تعداد فایلها به نسبت بسیار کم می باشند بنابراین مدیریت آنها بدون ابزار نیز بخوبی ممکن است.

طبق بررسی انجام شده تعداد کل فایلها (به غیر از فایلها ثابت مثل تصاویر، فایلهای layout و فایلهای سیستمی) در این دو معماری طبق جداول 4 و 5 باعث کاهش 97درصدی کل فایلها در معماری مطرح شده می شود.

نام لایه	فرمول MVC	فایل
View	m(JS+CSS+HTML)	900
کنترلر	m(CL)(0.9)	270
DAL و BL	m(DAL+BL)(0.1)	60
تعداد کل فایلهای برنامه		1230

جدول 4: محاسبه تعداد فایل در معماری MVC

در جدول 4 مقدار 0.9یعنی اکثر ماژولها در لایه کنترلر دارای فایل جداگانه هستند و 0.1 یعنی ماژولها در لایه های DAL و BL بسته به نوعشان دسته بندی شده و از هر 10 ماژول یک فایل تولید می شود.

نام لایه	فرمول SP-Based	فایل	کاهش
View	m(JS+CSS+HTML)/30	30	٪97
کنترلر	(m/30)+1	11	٪96
DAL و BL	ثابت	0	0
تعداد کل فایلهای برنامه		41	٪97

جدول 5: محاسبه تعداد فایل در معماری SP-Based

در جدول 5 مقدار 30 یعنی تعداد 30 ماژول برای نمایش بهینه در هر صفحه HTML مناسب است.

5-3- هزینه و سرعت توسعه

هزینه و سرعت توسعه بر اساس تعداد خطوط و سرعت کدنویسی محاسبه می شود. سرعت بسته به نوع کد و میزان پیچیدگی آن متغیر میباشد.

تعیین سرعت کدنویسی طبق تخمین صورت گرفته از شرایط اجرای پروژه در هر دو معماری و میزان تخصص نیروی انسانی در لایه های مختلف طبق جدول 6 و 7 ارزیابی شده است. سرعت براساس واحد خط/ساعت برای یک نفر می باشد. مثلا یک نفر در لایه View در معماری MVC .net core، 30 خط می تواند در یک ساعت تولید کند. در نهایت معماری مطرح شده باعث کاهش 82درصدی هزینه اجرای پروژه شده است.

نام لایه	سرعت	خطوط	زمان (ساعت)
View	30	37500	1250
کنترلر	15	11200	746
DALو BL	15	17600	1173
بانک	10	16000	1600
جمع		82300	4769

جدول 6: محاسبه زمان اجرای پروژه در معماری MVC

لایه	سرعت	خطوط	زمان (ساعت)	کاهش
View	10	4980	498	٪60
کنترلر	10	200	20	٪97
DAL و BL	---	---	---	--
بانک	15	4800	320	٪80
جمع		9980	818	٪82

جدول 7: محاسبه زمان اجرای پروژه در معماری SP-Based

5-4- اجرای بهینه درخواست ها

اساس کار در یک برنامه کلاینت سروری مدیریت بهینه درخواست ها می باشد. جدول 8 روی 5 شاخص مربوط به درخواست ها یک مقایسه بین دو معماری مطرح انجام داده است.

شاخص	MVC	SP-Based
چک سطح دسترسی	پایین (قراردادن هارد کد در اکشن ها)	بالا (تنها با یک کوئری)
اعتبارسنجی ورودی	پایین (تعیین نوع فیلدها با هارد کد در مدل مربوطه)	بالا (چک کردن پارامترهای ورودی به صورت خودکار)
هندل کردن خطاها	پایین (ساختارهای کنترل خطا باید تکرار شود)	بالا (قراردادن ساختار کنترل خطا در چند جای معدود)
لاگ کردن درخواستها	پایین (ساختار ثبت جزئیات درخواست، باید تکرار شود)	بالا (درخواست ها از یک جا عبور می کنند یک بار نوشته می شود)
درخواستهای همزمان	پایین (پیاده سازی آن پیچیده است)	بالا (پیاده سازی آن ساده و منعطف است)

جدول 8: مقایسه نحوه برخورد دو معماری با درخواست های کلاینت

5-5- کاهش شدید ترافیک شبکه و سرعت تبادل دیتا

در سامانه های بزرگ اجرای بهینه درخواست با کمترین بار ترافیکی و حداکثر سرعت هدفی است که به هر چه به آن نزدیکتر شوند باز هم تمایل دارند تا بهینه سازی را بیشتر ادامه دهند. در جدول 9 دلایل بهینه بودن روش SP-Based نسبت به MVC را با شاخص های مرتبط بررسی شده است.

شاخص	MVC	SP-Based
ترافیک ارسال درخواست	بالا (باید پارامترها در قالب مدل ارسال شود)	پایین (خودکار و عدم نیاز به مدل سازی)
ترافیک لایه کنترلر و بانک	بالا (وجود بیشتر دستورات SQL در لایه کنترلر)	پایین (اجرا دستورات SQL در SP)
ترافیک نتیجه درخواست	بالا (نتیجه در قالب مدل برگشت داده می شود)	پایین (خودکار و عدم نیاز به مدل سازی)

بهره وری در کشینگ داده	پایین (انعطاف پایین در بروزرسانی کش)	بالا (قابلیت بالا در بروزرسانی کش)

جدول 9: مقایسه نحوه برخورد دو معماری با ترافیک شبکه و تبادل دیتا

5-6- کپسوله کردن لایه کسب و کار

از آنجا که منطق کسب و کار در SP-Based در رویه های ذخیره شده قرار دارد باعث مزیت امکان محدود کردن دسترسی اعضاء پروژه به منطق کسب و کار و عدم دستکاری ناخواسته و افزایش امنیت پروژه شده است. بهبودهای صورت گرفته در این حوزه در جدول 10 بررسی شده است.

شاخص	MVC	SP-Based
هزینه انتشار ورژن جدید	بالا (با یک تغییر کوچک در منطق باید کل پروژه کامپایل گردد)	پایین (به خاطر قرار داشتن منطق در رویه های دخیره شده)
اثرات جانبی کدها	بالا (به خاطر ارتباط و وابستگی زیاد لایه ها با یکدیگر)	پایین (به خاطر ماهیت ایزوله بودن رویه های ذخیره شده)
امنیت دسترسی به سورس کد	پایین (تعیین دسترسی در حد یک عملیات بسیار مشکل است)	بالا (براحتی می توان دسترسی به سورس عملیاتها را تعیین کرد)

جدول 10: مقایسه نحوه برخورد دو معماری با کدنویسی در سطح کنترلر

6- نتیجه گیری

فرض کنید معماری حرفه ای هستید و پیشنهاد ساخت یک برج 60 طبقه با کاربری های متعدد در بهترین نقطه شهر به شما پیشنهاد شده است و کارفرما از شما خواسته با کمترین هزینه، بهترین کیفیت را ارائه دهید و در ضمن در کوتاهترین زمان پروژه را تحویل دهید. در چنین شرایطی باید نقشه و معماری را طراحی و پیاده سازی کنید که همزمان به تمام این اهداف برسد.

همین شرایط را در یک پروژه نرم افزاری تصور کنید یک شرکت بین المللی با شعب متعدد در سراسر دنیا با مشتریان فراوان به دنبال پیاده سازی سامانه ای است، که تمام عملیات فروش، بازاریابی، تحقیق و توسعه، پرسنلی و ... آن را یکپارچه در یک سامانه سامان دهی نماید. در چنین شرایطی باید به دنبال معماری باشید که بتواند جوابگوی این حجم بزرگ عملیات باشد و توامان هزینه را کاهش داده و پیاده سازی را در کمترین زمان انجام دهد.

معماری SP-Based با هدف تسهیل پیاده سازی پروژه های بزرگ گام های بزرگی برداشته و برای این منظور حجم کدنویسی را بیش از 80درصد کاهش داده است. بدین ترتیب سرعت و هزینه اجرای پروژه، کاهش چشم گیری پیدا نموده است. آینده دنیای نرم افزار با توجه به شتاب بالا در افزایش کاربران آنلاین، یکپارچه کردن سیستم های جزیره ای و حداکثر کردن استفاده بهینه از سخت افزار و کاهش هزینه ها نیاز به رویکردی جدید با فراهم کردن مزایای گفته شده دارد.

مراجع

[1] A Complete Survey on Software Architectural Styles and Patterns
[2] https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html
[3] https://searchmicroservices.techtarget.com/definition/event-driven-architecture-EDA
[4] https://whatis.techtarget.com/definition/engine