دانشگاه شهید بهشتی

دانشکده مهندسی و علوم کامپیوتر

کاهش شروع‌های سرد در پلتفرم‌های بدون سرور

گزارش سمینار کارشناسی ارشد مهندسی کامپیوتر
گرایش نرم‌افزار

نگارش

امیرمحمد کرم‌زاده

استاد راهنما

دکتر علیرضا شاملی

خرداد ۱۴۰۰

چکیده

لورم ایپسوم متن ساختگی با تولید سادگی نامفهوم از صنعت چاپ و با استفاده از طراحان گرافیک است. چاپگرها و متون بلکه روزنامه و مجله در ستون و سطرآنچنان که لازم است و برای شرایط فعلی تکنولوژی مورد نیاز و کاربردهای متنوع با هدف بهبود ابزارهای کاربردی می باشد. کتابهای زیادی در شصت و سه درصد گذشته، حال و آینده شناخت فراوان جامعه و متخصصان را می طلبد تا با نرم افزارها شناخت بیشتری را برای طراحان رایانه ای علی الخصوص طراحان خلاقی و فرهنگ پیشرو در زبان فارسی ایجاد کرد. در این صورت می توان امید داشت که تمام و دشواری موجود در ارائه راهکارها و شرایط سخت تایپ به پایان رسد وزمان مورد نیاز شامل حروفچینی دستاوردهای اصلی و جوابگوی سوالات پیوسته اهل دنیای موجود طراحی اساسا مورد استفاده قرار گیرد.

واژگان کلیدی: رایانش ابری، رایانش بدون سرور، شروع‌های سرد، function-as-a-service، FaaS

فهرست مطالب

فهرست شکل‌ها

فهرست جداول

فصل ۱

مقدمه

۱.۱ صورت مسئله

چه روش‌هایی برای کاهش تعداد شروع‌های سرد در پلتفرم‌های بدون‌سرور [۱] با حداقل سربار [۲] و زمان اجرایی وجود دارند؟

۲.۱ انگیزه‌ی تحقیق

رایانش بدون سرور[۳] یکی از مسائل داغ و محبوب این‌روزهای دنیای مهندسی نرم‌افزار و رایانش ابری است. رایانش بدون سرور حوزه‌ی جدیدی را در توسعه‌ی محصول و استقرار[۴] اپلیکیشن‌ها باز کرده است. یکی از دلایل محبوبیت استفاده از پلتفرم‌های بدون‌سرور و تمایل توسعه‌دهندگان برای مهاجرت به سمت آن، استفاده بیش از پیش از معماری میکروسرویس و نانوسرویس در توسعه‌ی محصولات و حرکت معماران و مهندسین نرم‌افزار در تولید و مهاجرت برنامه‌های کاربردی با این معماری‌ها است.

از دید توسعه دهنده، رایانش ابری با حذف دخالت مستقیم کاربران انتهایی در مدیریت زیرساخت ازجمله lrauto-scaling یا lrload-balancing موجب بهبود سرعت توسعه محصول و تمرکز کاربران برروی منطق[۵] برنامه است. همچنین، برای علاوه‌بر آسانی استفاده و پنهان‌سازی پیچیدگی مدیریت سرور از کاربر، به علت اینکه ارائه‌دهندگان خدمات ابری در نقاط مختلف جهان حضور دارند و همچنین کانفیگ بهینه CDNها؛ ارتباطات بین سرورها و کاربران با حداقل تاخیر[۶] صورت می‌گیرد.

به طور کلی، یک پلتفرم بدون‌سرور را هر پلتفرم محاسباتی تعریف کرد که در آن مدیریت مستقیم سرور از کاربران مخفی شده و برنامه‌های کاربردی به صورت اتوماتیک در آن مقیاس‌پذیر می‌شوند و تنها هنگامی که در حال استفاده از پلتفرم هستیم، هزینه آن را پرداخت می‌کنیم. [۱]

یکی از قابلیت‌هایی که در رایانش بدون سرور باعث محبوبیت آن شده‌است، قابلیت Scale-to-Zero است. این بدان معنی است که هنگامی که از یک کانتینر استفاده‌ای نداریم، منابع آن گرفته می‌شوند و کانتینر اصطلاحا

[1] serverless
[2] Overhead
[3] Serverless Computing
[4] Deployment
[5] logic
[6] Latency

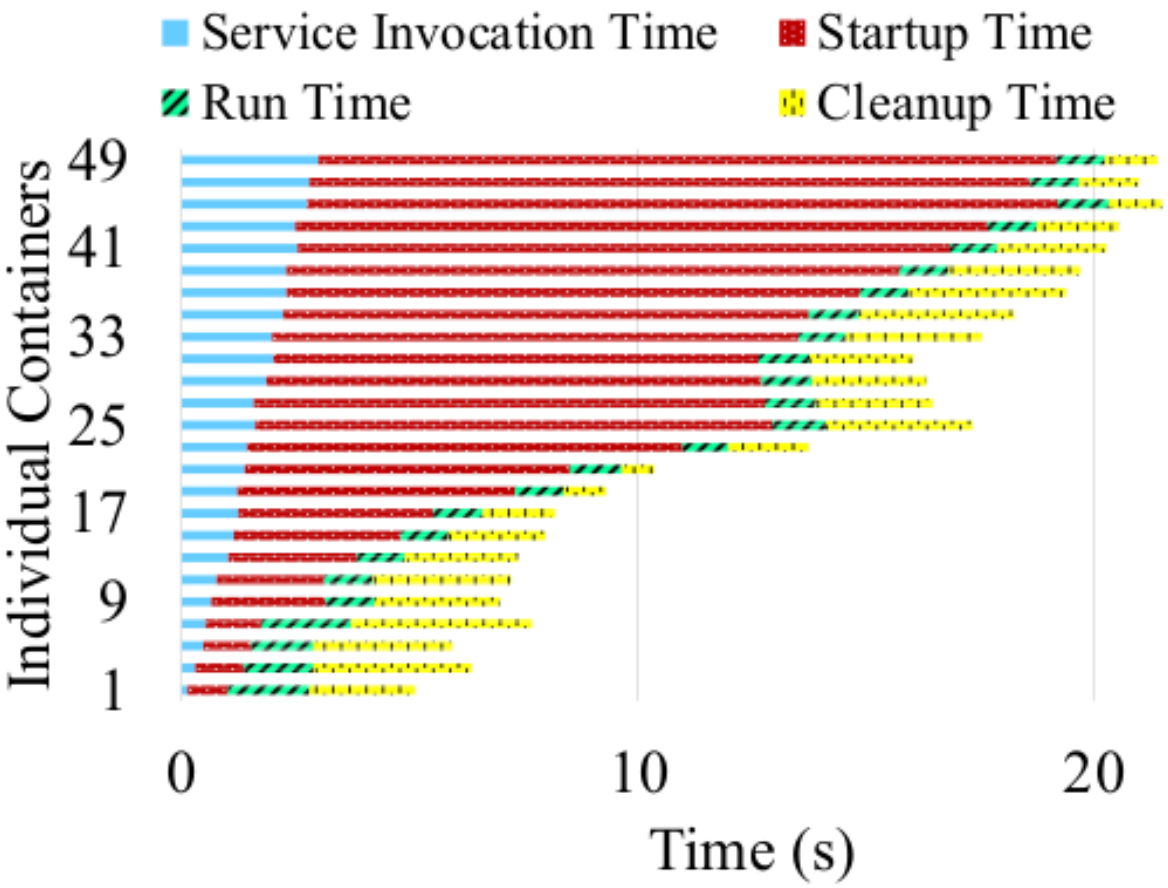
Zero-Scaled می‌شود. این خود موجب قابلیت پرداخت تنها در حین مصرف ما از تابع می‌شود. اما مشکل اصلی زمانی است که درخواست جدیدی برای کانینر Zero-Scale شده می‌رسد؛ در این حالت باید درخواست منتظر مانده تا سلسله‌ای از آماده‌سازی ها انجام شوند تا کانتینر مربوطه مجددا اجرا شود. این خود باعث تاخیری مضاف برای پاسخ‌دهی به درخواست را موجب می‌شود که به این تاخیر مشکل شروع‌سرد [1] گفته می‌شود. در واقع می‌توان گفت تاخیر شروع سرد ناشی از تلاش ما در تعادل بین تاخیر در پاسخگویی به درخواست‌ها و هزینه (هزینه‌های استفاده از رم و سی‌پی‌یو و ...) است.

متاسفانه در سالیان اخیر و در عین داغ‌بودن مبحث و نیاز بازار به حل این مشکل، این مشکل چندان در محیط‌های آکادمیک مورد بررسی قرار نگرفته است. البته در نگاه کلی‌تر، مشکلات و مسائل بار مربوط به رایانش بدون‌سرور، اکثرا در محیط‌های آکادمیکی مثل دانشگاه‌ها با کم محلی روبرو شده‌اند. در این میان، پلتفرم‌های متن‌باز [2] که دارای جوامع بسیار گسترده‌ای نیز هستند، به خاطر این سری مسائل باز که شرکت‌های تجاری در حال صرف هزینه‌های هنگفتی برای حل و فصل مسکلات مربوط به آن هستند، به شدت از رقابت عقب مانده‌اند. انگیزه ما برای انجام این پژوهش در این است که اولا بتوانیم به راهکار مناسب‌تری برای حل مشکل مربوط به شروع سرد در پلتفرم‌های بدون سرور برسیم و ثانیا بتوانیم با مشارکت در بهبود یکی از پلتفرم‌های آزاد در توسعه این پلتفرم‌ها تاثیر کوچکی داشته باشیم.

۳.۱ مثال مرتبط

در مقاله [۲] آقای لین و همکارش توانستند تا با اسفاده از استخر گرم و نگه‌داری کانتینر توابعی که محبوبیت استفاده دارند، مدت زمان پاسخ را در حدود ٪۸۵ کاهش دهند. این بهبود در پلتفرم knative اجرا شد و ایده‌ی مقالات دیگری نیز بوده است.

[1] Cold Start
[2] Open Source



شکل ۱.۱: اهمیت شروع‌سرد

۴.۱ اهمیت موضوع

اگرچه پلتفرم‌های بدون‌سرور از نظر هزینه، scaling، راحتی استفاده و گستردگی پوشش جغرافیایی برای ما بهینه هستند؛ اما مشکل شروع سرد مشکلی نیست که بتوان به سادگی از آن گذر کرد. تصویر ۱.۱ که از [۳] گرفته شده است، نشان می‌دهد که بیش از ٪۸۰ زمان اجرای کامل یک کانتینر در پلتفرم‌های بدون سرور به آماده‌سازی آن یا شروع سرد اولیه[1] مربوط می‌شود.

ستون قرمز رنگ زمان آماده سازی کانتینر را نمایش می‌دهد. این زمان همان زمان شروع سرد است که دلایل مختلفی از جمله آماده‌سازی کانتینر یا حضور در صف انتظار برای تخصیص منابع باشد. بنابراین، با به‌کارگیری یک استراتژی مناسب می‌توان این زمان را به حداقل رسانید.

متاسفانه تاخیر شروع سرد باعث شده تا توسعه‌دهندگان اقبال کمتری به استفاده از پلتفرم‌های بدون سرور

[1] First Cold Start

داشته باشند. به گونه‌ای که از بین ۱۰۰۰ برنامه‌ی کاربردی بزرگ در پلتفرم Microsoft Azure، تنها یک مورد مربوط به یک برنامه تجاری باشد [۴]. این موضوع نشان می‌دهد که علی‌رغم پتانسیل بالای رایانش بدون سرور، وجود مشکلات جدی ازجمله شروع سرد، باعث امتناع توسعه‌دهندگان از مهاجرت به پلتفرم‌های بدون سرور باشد.

۵.۱ نتیجه‌های مهم تحقیق

لورم ایپسوم متن ساختگی با تولید سادگی نامفهوم از صنعت چاپ، و با استفاده از طراحان گرافیک است، چاپگرها و متون بلکه روزنامه و مجله در ستون و سطرآنچنان که لازم است، و برای شرایط فعلی تکنولوژی مورد نیاز، و کاربردهای متنوع با هدف بهبود ابزارهای کاربردی می باشد، کتابهای زیادی در شصت و سه درصد گذشته حال و آینده، شناخت فراوان جامعه و متخصصان را می طلبد، تا با نرم افزارها شناخت بیشتری را برای طراحان رایانه ای علی الخصوص طراحان خلاقی، و فرهنگ پیشرو در زبان فارسی ایجاد کرد، در این صورت می توان امید داشت که تمام و دشواری موجود در ارائه راهکارها، و شرایط سخت تایپ به پایان رسد و زمان مورد نیاز شامل حروفچینی دستاوردهای اصلی، و جوابگوی سوالات پیوسته اهل دنیای موجود طراحی اساسا مورد استفاده قرار گیرد.

ساختار این گزارش به این ترتیب خواهد بود:

درادبیات موضوع مروری بر واژگان، مفاهیم تخصصی و هر آن‌چه که در ادامه به آن نیاز پیدا خواهیم کرد، خواهیم داشت. سپس در فصل کارهای مرتبط به بیان مشروح تحقیقات انجام شده خواهیم پرداخت و در نتیجه‌گیری و کارهای آینده خلاصه‌ای از مطالعات انجام شده و مسائل باز خواهیم پرداخت.

فصل ۲

مروری بر ادبیات

در این بخش سعی داریم تا با مروری بر اصطلاحات و ابزارهای مورد استفاده در پژوهش‌های بررسی شده، با پیش‌نیازهای مبحث موردنظر آشنا شویم.

۱.۲ رایانش بدون سرور

رایانش بدون سرور[1] در سال ۲۰۱۴ توسط شرکت آمازون برای اولین بار معرفی شد. تا قبل از این رایانش بدون سرور یک مفهوم انتزاعی [2] در شبکه بود که شرکت آمازون با ارائه پلتفرم AWS Lambda Functions به معرفی آن پرداخت. سپس در سال ۲۰۱۶ سایر ارائه دهندگان خدمات ابری نیز به ارائه پلتفرم‌های بدون سرور خود پرداختند. در این سال به ترتیب شرکت‌های گوگل پلتفرم google cloud functions یا به اختصار GCP، شرکت مایکروسافت پلتفرم Microsoft Azure functions و شرکت IBM به معرفی IBM OpenWhisk پرداختند. البته باید توجه داشت که مفهوم رایانش بدون سرور به طور کامل توسط ارائه دهندگاه خدمات ابری پیاده‌سازی نشده است و جای کار بسیاری دارد (با مطالعه این گزارش به مرور متوجه نواقص موجود خواهید شد).

در رایانش بدون سرور ما از نقطه قوت ماشین‌های مجازی که ایزولاسیون برنامه‌های مختلف از همدیگر بود استفاده کرده‌ایم. منتها این مورد را با مفهوم کانتینر ها پیاده‌کرده ایم. در ادامه راجع به کانتینرها نیز بحث خواهیم کرد.

۱.۱.۲ تعریف رایانش بدون سرور

رایانش بدون سرور مبحثی از رایانش ابری است که در آن بحث مدیریت حافظه یا Storage، مدیریت زیرساخت و بحث‌های networking با انتزاع بالایی به مصرف کاربر می‌رسد. به عبارت دیگر، تمامی مدیریت ای بخش‌ها بر عهده ارائه دهندگان است و ما اصلا با این بحث ها سروکاری نداریم. در واقع، هدف اصلی رایانش بدون سرور هم این است که این پیچیدگی‌ها را از کاربر بگیرد.

به طور کلی، یک پلتفرم بدون‌سرور را هر پلتفرم محاسباتی تعریف کرد که در آن مدیریت مستقیم سرور از کاربران مخفی شده و برنامه‌های کاربردی به صورت اتوماتیک در آن مقیاس‌پذیر می‌شوند و تنها هنگامی که در

[1] Serverless Computing
[2] abstract

حال استفاده از پلتفرم هستیم، هزینه آن را پرداخت می‌کنیم. [۱]

بسیاری از افراد، serverless و faas را معادل یک‌دیگر می‌دانند درحالی‌که اصلا این‌گونه نیست. در ادامه راجع به این بحث به طور مفصلی بحث خواهیم کرد اما باید بدانیم که این دو مقوله کاملا جدا از همدگیر هستند و مجددا تاکید می‌کنیم که رایانش بدون سرور یک مدل اجرایی در رایانش ابری است.

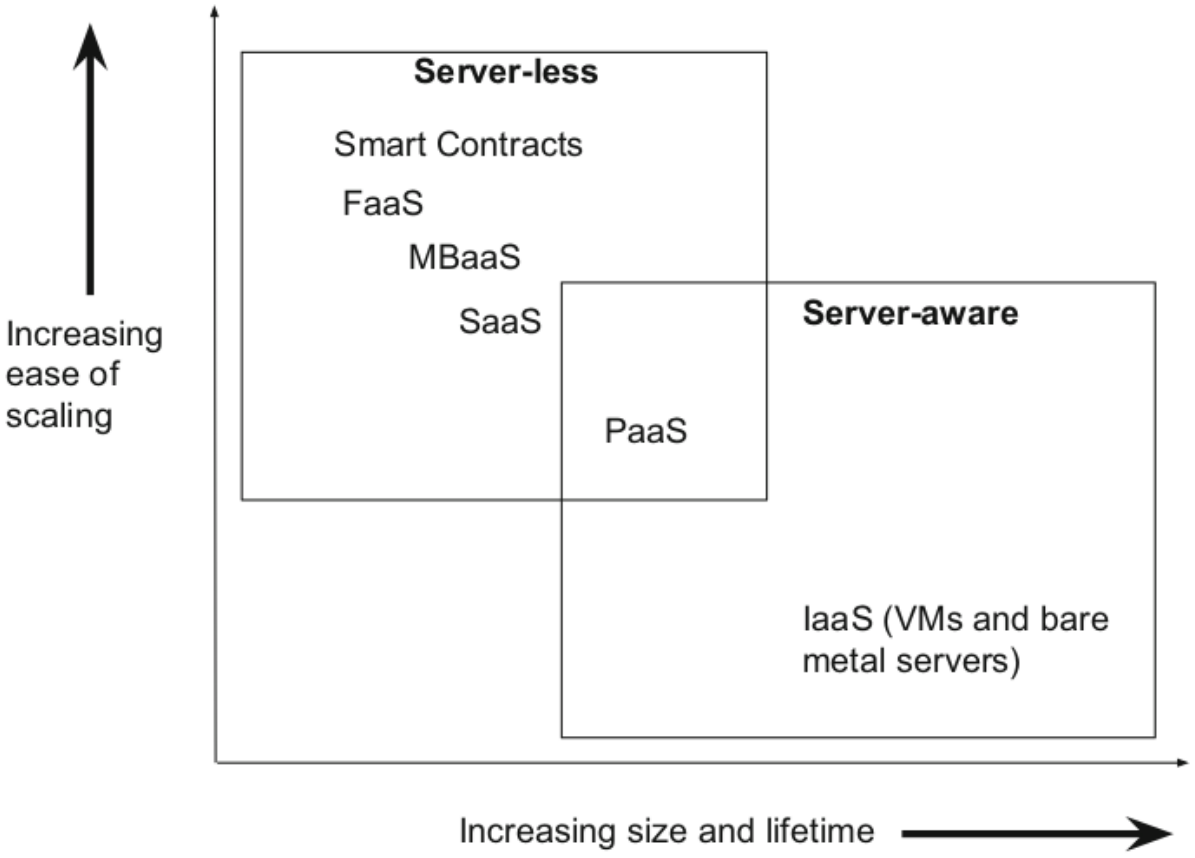
ازطرفی رایانش بدون سرور را باید نقطه مقابل رایانش سرور آگاهانه [۱] دانست که در آن از اطلاعات سرور در اختیار گرفته کاملا اگاهیم، کاملا بر مدیریت آن اشراف داریم و هرگونه تغییر از جمله متعادل‌سازی بارها، auto-scaling و ... باید توسط کاربر انجام شود.

یک مثال از پیاده‌سازی رایانش سرور آگاهانه را در زیرساخت به عنوان سرویس [۲] یا به اختصار IaaS است. در نقطه مقابل در رایانش بدون سرور هیچ کنترلی بر روی سرور نداریم، تنها می‌توانیم یک برنامه را بر روی سرور اجرا کنیم یا اجرای آن را به حالت تعلیق درآورده یا آن را از روی سرور حذف کنیم که هیچ‌کدما از این موارد نیز به صورت مستقیم انجام نمی‌گیرد؛ بلکه رابط گرافیکی و API وجود دارد که از طریق آن‌ها این تغییرات را اعمال می‌کنیم. بنابراین در رایانش بدون سرور، عملا هیچ راهی برای مدیریت مستقیم سرور و زیرساخت نداریم.

شکل ۱.۲ مرزهای بین رایانش بدون‌سرور و رایانش سرور آگاهانه را نمایش می‌دهد.

البته باید به این نکته توجه‌داشت که امروزه مرزهای بین رایانش سرور آگاهانه با رایانش بدون سرور در حال کمرنگ شدن و بعضا از بین رفتن است و این تقسیم بندی ابدا قاطعیت ندارد.همچنین تفکیک برخی موارد مانند Platform-as-a-Serice یا به اختصار PaaS به راحتی انجام نمی‌گیرد بلکه این نوع رایانش می‌تواند از نوع باسرور یا بدون سرور باشد. در این شکل هرچه به سمت محور افقی حرکت می‌کنیم دانه‌بندی و طول‌عمر افزایش پیدا می‌کند و هرچه به سمت بالاتر می‌رویم، scaling راحت‌تر انجام می‌گیرد.

[1] Server Aware
[2] Infrastructure as a service



شکل ۱.۲: مرزهای رایانش بدون سرور و رایانش سرورآگاهانه

فصل ۳

کارهای مرتبط

لورم ایپسوم متن ساختگی با تولید سادگی نامفهوم از صنعت چاپ، و با استفاده از طراحان گرافیک است، چاپگرها و متون بلکه روزنامه و مجله در ستون و سطرآنچنان که لازم است، و برای شرایط فعلی تکنولوژی مورد نیاز، و کاربردهای متنوع با هدف بهبود ابزارهای کاربردی می باشد، کتابهای زیادی در شصت و سه درصد گذشته حال و آینده، شناخت فراوان جامعه و متخصصان را می طلبد، تا با نرم افزارها شناخت بیشتری را برای طراحان رایانه ای علی الخصوص طراحان خلاقی، و فرهنگ پیشرو در زبان فارسی ایجاد کرد، در این صورت می توان امید داشت که تمام و دشواری موجود در ارائه راهکارها، و شرایط سخت تایپ به پایان رسد و زمان مورد نیاز شامل حروفچینی دستاوردهای اصلی، و جوابگوی سوالات پیوسته اهل دنیای موجود طراحی اساسا مورد استفاده قرار گیرد.

فصل ۴

نتیجه‌گیری

در فصل قبل در ارتباط با راهکارهای ارائه شده توسط هر یک از مقاله‌ها به تفصیل، تشریح نمودیم. در این فصل می‌خواهیم به بیان نتایج و مقایسه راهکارهای ارائه شده بپردازیم. در ابتدا به مقایسه‌های روش‌های مبتنی بر مکانیزم DRX/DTX با یکدیگر می‌پردازیم.

مراجع

[1] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "The rise of serverless computing," *Commun. ACM*, vol.62, p.44-54, Nov. 2019.

[2] P.-M. Lin and A. Glikson, "Mitigating cold starts in serverless platforms: A pool-based approach," *arXiv preprint arXiv:1903.12221*, 2019.

[3] A. Mohan, H. Sane, K. Doshi, S. Edupuganti, N. Nayak, and V. Sukhomlinov, "Agile cold starts for scalable serverless," in *11th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 19)*, 2019.

[4] M. Shahrad, R. Fonseca, I. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini, "Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pp.205-218, 2020.