به نام خدا

گزارش پروژه نهایی

شبکه‌های کامپیوتری پیشرفته

سپند سراج

۸۱۰۱۰۱۲۱۱

سید سعید صفایی

۸۱۰۱۰۱۰۱۹

باقی‌مانده جمع شماره‌ها به عدد ۸: ۶

شکل‌های مربوط: شکل 6(B) و 6(C) مقاله‌ی ۱

مقدمه

زنجیره سرویس‌ها در شبکه‌های کامپیوتری به مجموعه‌ای از سرویس‌ها گفته می‌شود که یک میزبان در زمان دریافت یک پکت از شبکه روی آن انجام می‌دهد تا بتواند آن را به درستی پردازش و سرویس‌دهی کند. از جمله سرویس‌های از این قبیل میتوان به Intrusion Detection System(IDS)، Firewall(FW) و Load Balancer(LB) اشاره کرد. این سرویس‌ها باید به صورت ترتیبی روی یک پکت اعمال شوند و به همین علت از نام زنجیره سرویس Service Function Chain(SFC) استفاده می‌کنیم.

به صورت تاریخی همیشه ارائه دهندگان سرویس‌های ابری که ارائه دهنده اصلی این سرویس‌ها بودند برای هر کدام از این سرویس‌ها از سخت‌افزارهای خاص منظوره و بسیار هزینه‌بر استفاده می‌کردند که قابلیت مقیاس پذیری بسیار کمی داشتند و نیازمند سرمایه‌گذاری اولیه بسیار زیاد بودند.

با پیشرفت تکنولوژی و افزایش چشم‌گیر کارایی پردازنده‌های عام‌منظوره مفهوم مجازی سازی Virtualizationتوجه سازندگان این سیستم‌ها را جلب کرد. از بزرگترین مزیت‌های مجازی‌سازی سیستم‌ها می‌توان به افزایش مقیاس‌پذیری و کاهش هزینه‌ها اشاره کرد.

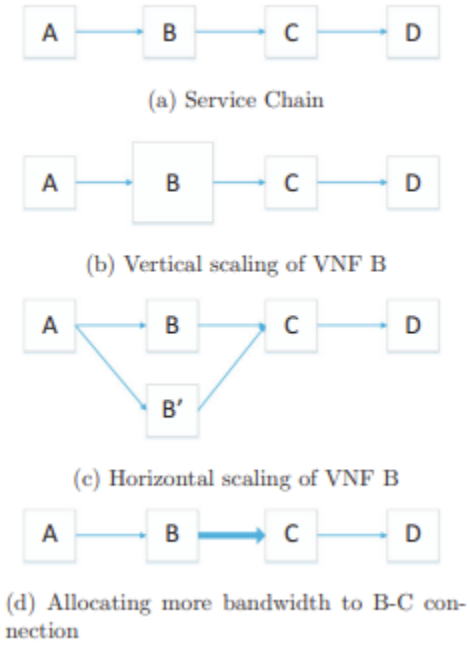
طراحی‌کنندگان این سیستم‌های مجازی برای تست و بررسی میزان کارایی نیازمند سیستمی جهت شبیه‌سازی طرح‌های خود بودند تا بتوانند قبل از پیاده‌سازی چنین سیستم‌های پیچیده و هزینه بری از کارایی سیستم خود اطمینان حاصل کنند. شبیه‌ساز CloudSim یکی از این شبیه‌سازها است که تمامی ابزار و پارامترهای مورد نیاز جهت مطالعه و بررسی کارایی سیستم‌های زنجیره سرویس شبکه‌ای را برای طراحی کنندگان فراهم می‌کند.

بخش ۰: مطالعه و بررسی مقاله

طوسی و همکاران [1] در سال ۲۰۱۹ مطالعه‌ای پیرامون تکنیک‌های مقیاس پذیری منابع برای زنجیره سرویس‌های شبکه‌ای ارائه انجام دادند. از دستاوردهای این مطالعه ارائه روشی نوین جهت مانیتورینگ و مقیاس‌پذیری لحظه‌ای منابع سیستم برای ارائه زنجیره سرویس‌های شبکه‌ای است. از نکات حائز اهمیت این مطالعه توجه به پارامترهای کارایی سیستم مانند تاخیر در پاسخگویی Latency است.

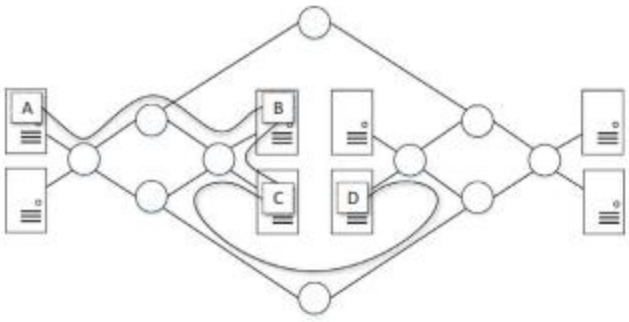
در طی این مطالعه فرض شده است که منابع اصلی که همواره مورد نیاز زنجیره سرویس‌دهی می‌باشند یکی پهنای باند Bandwidth و قدرت پردازش CPU می‌باشند. هرچند که الگوریتم‌های ارائه شده در این مطالعه قابل بسط دادن به دیگر منابع سیستمی نیز می‌باشند.

منابع فوق الذکر به دو طریق قابل مقیاس‌پذیری می‌باشند. اصطلاحا در مقیاس پذیری عمودی تعداد منابع ثابت است و کارایی آن‌ها افزایش می باشد اما در مقیاس‌پذیری افقی تعداد منابع افزایش می‌یابند. یکی دیگر از نکات حائز اهمیت در این مطالعه ارائه راهکارهای نوینی جهت پشتیبانی از هر دو نوع مقیاس‌پذیری به صورت لحظه‌ای در سیستم برای هر دو نوع منبع پهنای باند و قدرت پردازشی است.

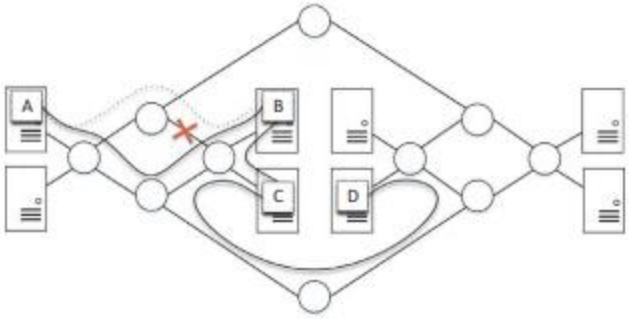


شکل ۱-۰.مکانیسم افزایش پهنای باند

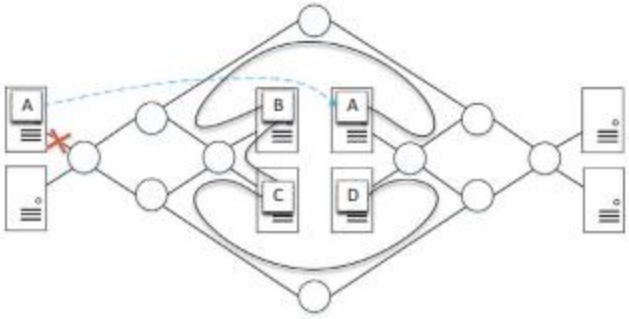
از دیگر تکنیک‌های ارائه شده در این مقاله جابجایی مکان سرویس‌دهی‌ها داخل شبکه است که بعضا در راستای رفع محدودیت‌های پهنای باند با استفاده از منابع فعلی کمک می‌کند. نحوه عملکرد این تکنیک را می‌توانید در شکل زیر مشاهده کنید:



(a) Service Chain Placement



(b) Flow Scheduling for Scaling Bandwidth



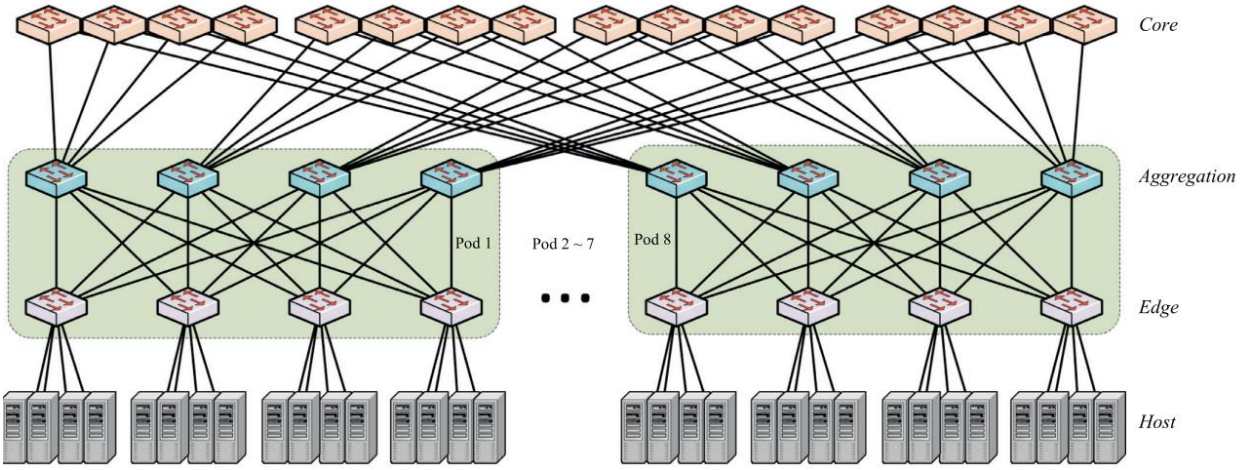
(c) VNF Migration for Scaling Bandwidth

شکل ۲-۰.نمونه‌ای از اجرای الگوریتم افزایش پهنای باند

راه‌اندازی آزمایش

در راستای صحت سنجی یافته‌های این مطالعه و راه‌حل پیشنهادی آزمایشی طراحی شده است که ما در این پروژه سعی در پیاده‌سازی آزمایشی با پارامترهای مشابه داریم.

ساختار فیزیکال این شبکه شبیه‌سازی شده از مدل fat tree 8 pod پیروی می‌کند که دارای ۳۲ سوییچ مرزی، ۳۲ سوییچ تجمیع کننده و ۱۶ سوییچ هسته‌ای است که مجموعا ۱۲۸ واحد پردازش کننده را به همدیگر متصل می‌کنند. ساختار پیشنهادی را می‌توانید در تصویر مشاهده کنید:



شکل ۳-۰.ساختار fat-tree شبکه توصیف شده در مقاله

در ادامه ویژگی‌های واحدهای پردازشی و انواع آنها تشریح شده‌اند که ما نیز محیط شبیه‌سازی خود را طبق همین جداول تنظیم کردیم.

Table 2
VM types for a 3-tier web application.

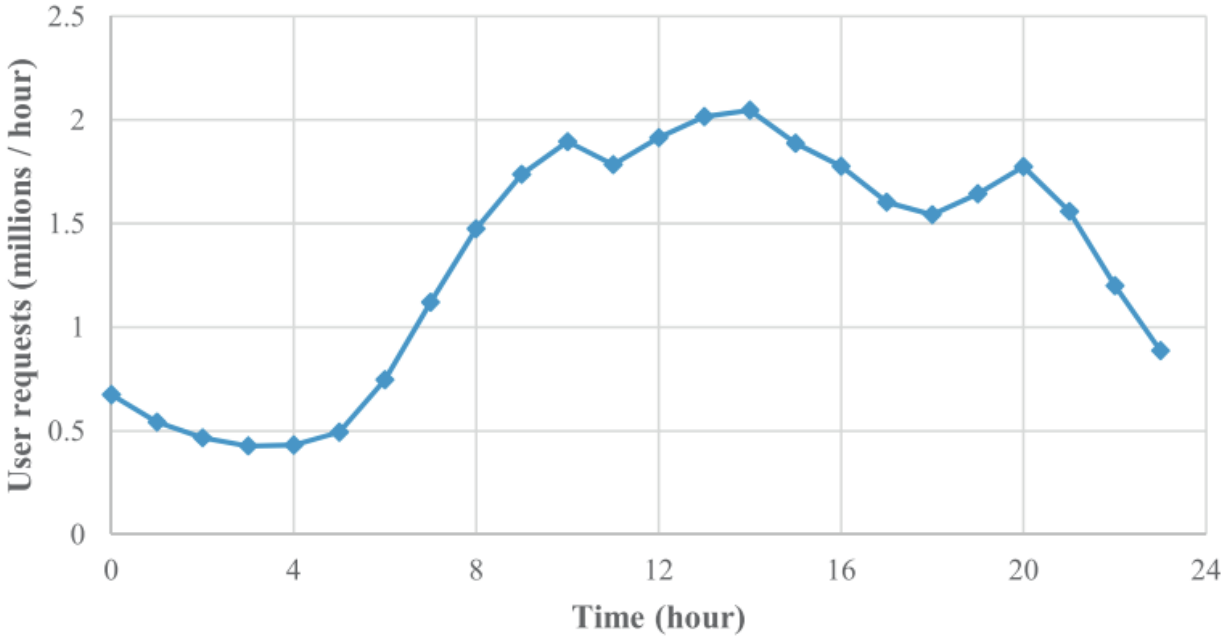
VM Type	CPU Capacity (cores*MIPS)	# of VMs
Web server	8*10,000	8
App server	4*10,000	24
Database	12*10,000	2

در ادامه نیز سیاست‌های سرورهای موجود در شبکه در برابر پکت‌های دریافتی از جهت زنجیره سرویس‌های ارائه شده توصیف شده‌اند.

Table 3
SFC policies defined for the 3-tier application in the evaluation.

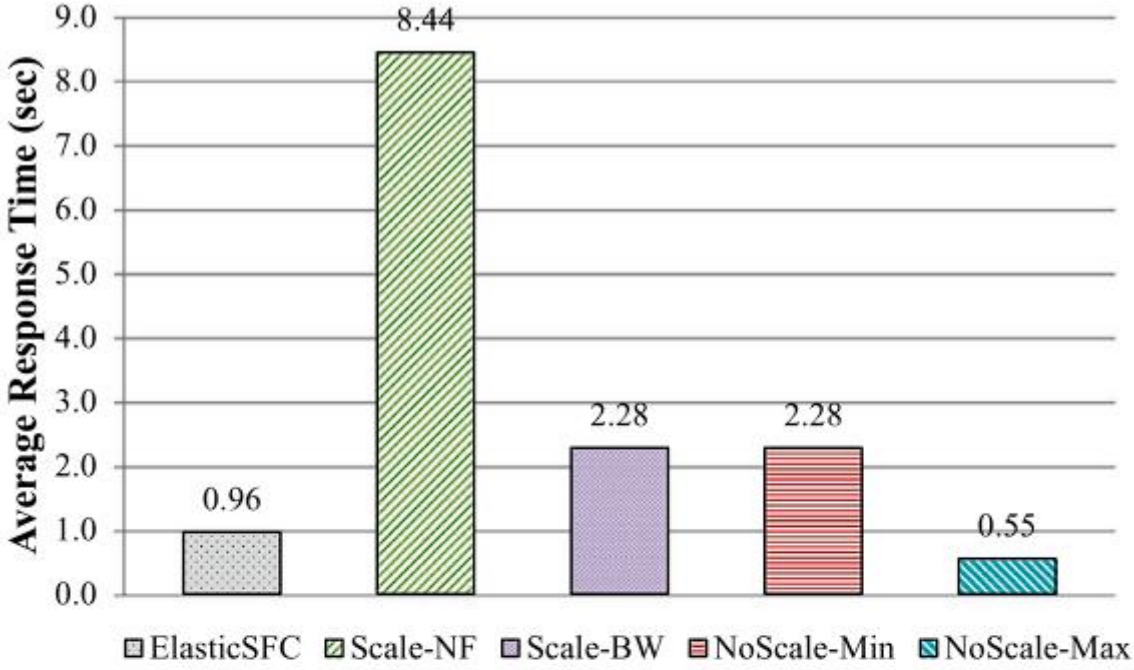
Source VM	Destination VM	SFC
Web server	App server	{FW, LB1}
App server	Database	{LB2, IDS}
Database	App server	{IDS, LB2}
App server	Web server	{LB1}

پس از طراحی سیستم مورد نظر در فضای شبیه‌سازی نوبت به طراحی یک سناریو نزدیک به واقعیت در راستای تست میزان کارایی سیستم می‌رسد. برای این بخش نویسندگان مقاله از الگوی ترافیک وبسایت ویکیپدیا به زبان آلمانی طی یک بازه ۲۴ ساعته استفاده کرده‌اند که در مجموع در این بازه ۲۹ میلیون درخواست به سرورها زده می‌شود. الگوی این ترافیک در شکل زیر مشاهده می‌شود.

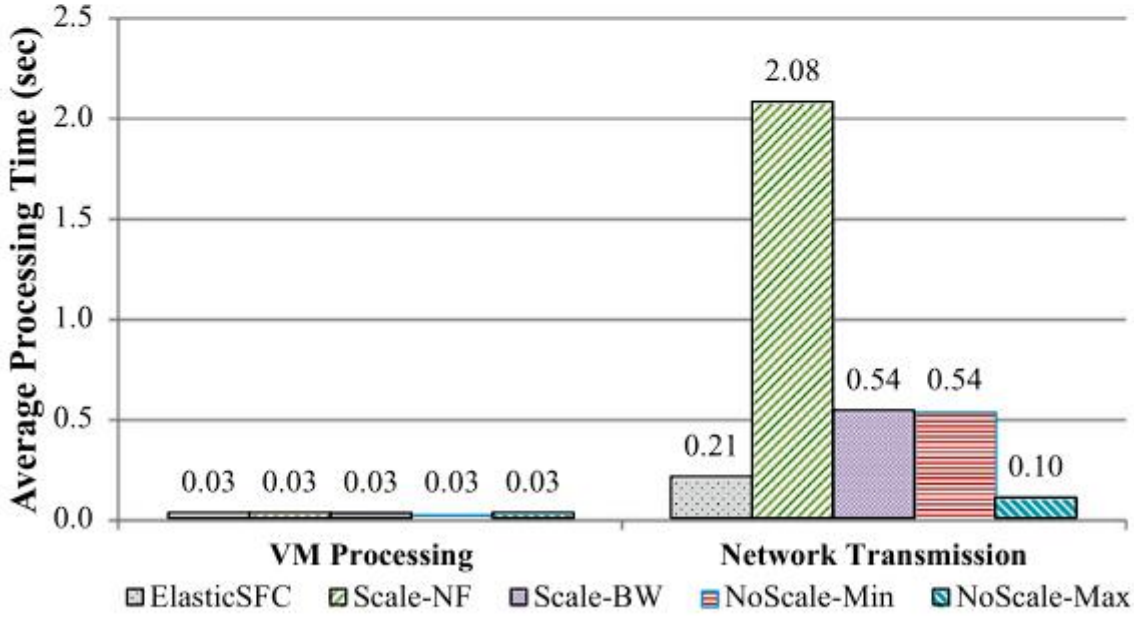


شکل ۴-۰.نمودار فراوانی درخواست‌ها در ورکلود ویکیپدیا

در انتها نویسندگان الگوریتم پیشنهادی خود را با ۴ الگوریتم پایه دیگر (حداقل منابع، حداکثر منابع، مقیاس پذیری پهنای‌باند و مقیاس پذیری قدرت پردازش) مقایسه کردند و به خروجی‌های زیر رسیدند. در بخش ۱ بیشتر راجع به آنها توضیح می‌دهیم و سعی می‌کنیم این خروجی‌ها را بازسازی کنیم.



(b) Average response time of requests.



(c) Average processing time within a VM and network transmissions.

شکل ۵-۰.مقایسه پارامترهای کارایی سیستم درسناریوهای مختلف

بخش ۱: ایجاد خروجی‌هایی مشابه مقاله

نصب Cloudsim-sdn

طبق توصیه تعریف پروژه نسخه cloudsimsdn که از شبیه‌ساز Cloudsim را انتخاب کردیم.

برای نصب این نسخه از شبیه ساز ابتدا سورس کد پروژه Cloudsim نسخه ۴ را از لینک [۲] دانلود می‌کنیم. همچنین Cloudsim-sdn را از لینک [۳] دانلود می‌کنیم. سپس فولدر

cloudsim-cloudsim-4.0\modules\cloudsim\src\main

از پروژه Cloudsim را داخل آدرس زیر از پروژه Cloudsim-sdn کپی می‌کنیم.

cloudsimsdn-master\src\main

در این مرحله پروژه cloudsim-sdn کامل است. و دیگر کاری با cloudsim نداریم.

در ادامه باید فایل pom.xml را بروزرسانی کنیم و خط زیر را در پایان وابستگی‌ها اضافه می‌کنیم.

```
<dependency>
    <groupId>com.opencsv</groupId>
    <artifactId>opencsv</artifactId>
    <version>3.7</version>
</dependency>
```

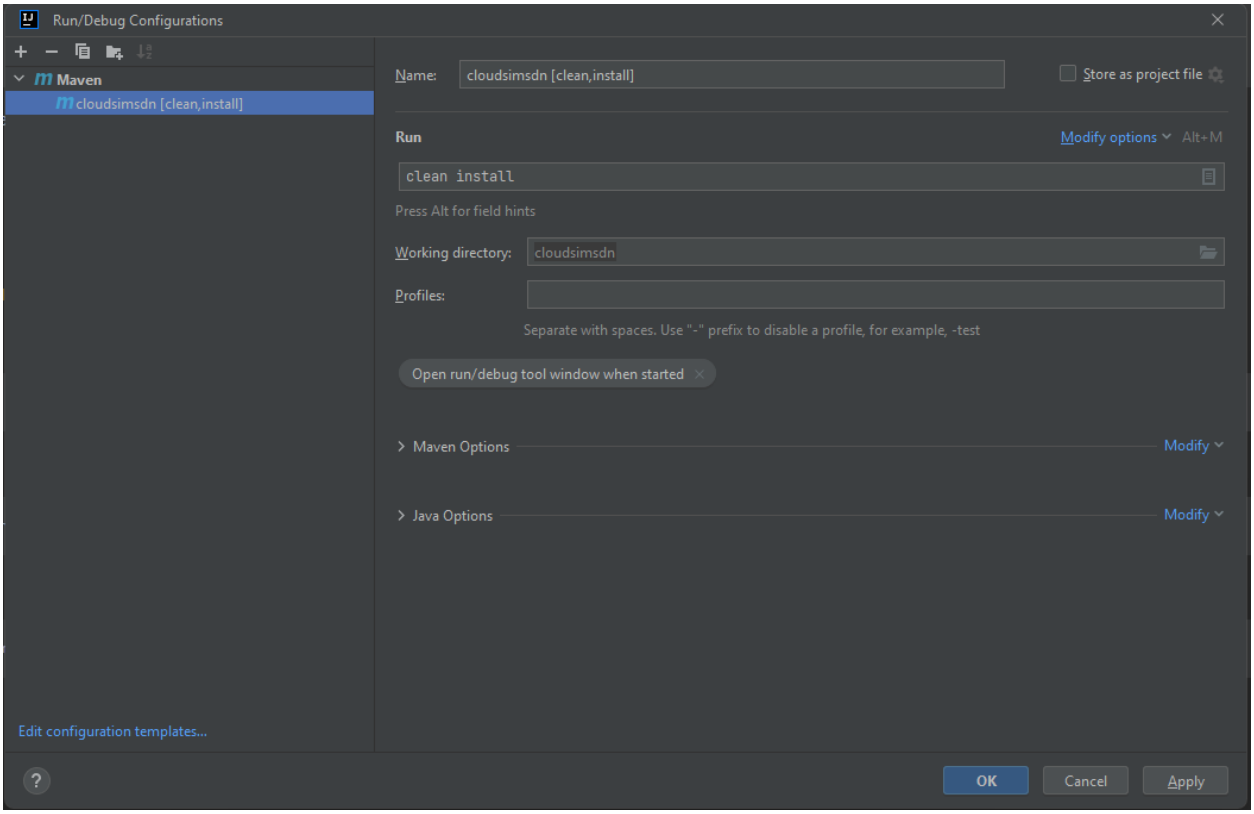
در این مرحله می‌توانیم به راه‌اندازی محیط توسعه بپردازیم.

ما در ادامه از کد منبعی که در فایل‌های کنار پروژه قرار داده‌ایم استفاده می‌کنیم. برخی از فایل‌ها در کلودسیم را تغییر دادیم تا بتوانیم به خواسته‌های پروژه برسیم. تغییرات مهم در ادامه‌ی این گزارش مستند شده‌اند.

راه اندازی محیط توسعه

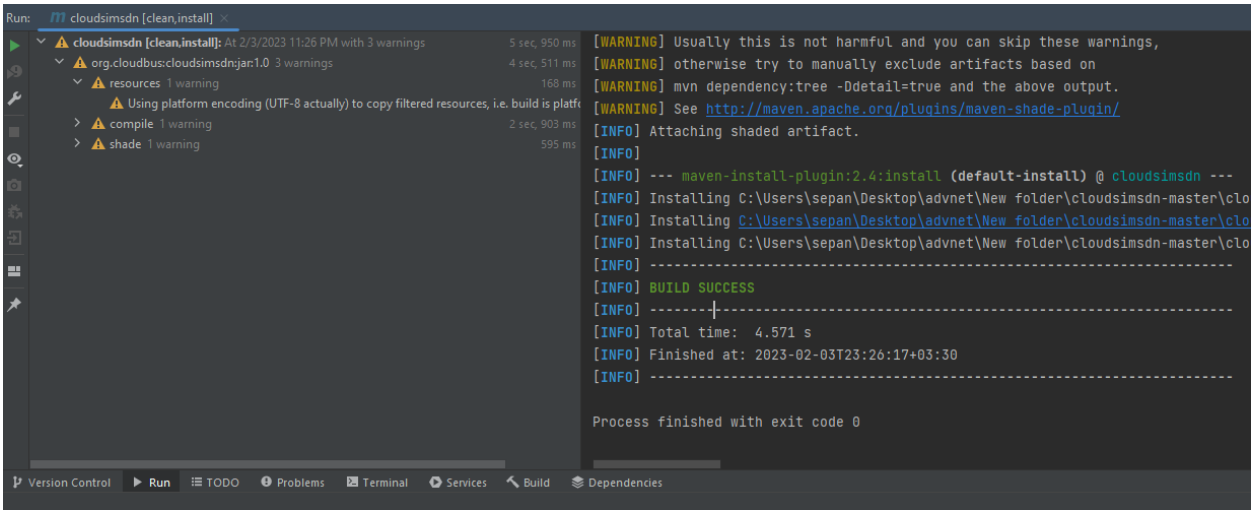
در این پروژه برای توسعه پروژه از نرم‌افزار IntelliJ IDEA Community edition استفاده می‌کنیم.

پس از افزودن پوشه پروژه به برنامه باید از طریق ابزار maven پروژه را نصب کنیم. برای انجام این کار ابتدا یک پروفایل اجرا با مشخصات زیر می‌سازیم.



شکل ۱-۰.آرگومان‌های لازم برای نصب برنامه

پس از افزودن با اجرای این دستور با موفقیت پروژه روی سیستم شما نصب می‌شود.



شکل ۲-۰. نصب و راه‌اندازی اولیه شبیه‌ساز

ایجاد توپولوژی فیزیکی شبکه

برای ایجاد توپولوژی فیزیکی شبکه، از فایل PhysicalTopologyGeneratorSFC موجود در پوشه‌ی

cloudsimsdn\src\main\java\org\cloudbus\cloudsim\sdn\example\topogenerators

استفاده می‌کنیم. توپولوژی‌ای که در مقاله استفاده کرده است، به طور کامل با تنظیمات این فایل مطابقت دارد؛ پس تنها لازم است که اجرایش کنیم تا فایل sfc.physical.fattree.json ایجاد شود. فایل حاصل را به پوشه‌ی

cloudsimsdn\project-acn\resources

می‌بریم.

ایجاد توپولوژی VMها

برای ایجاد توپولوژی VMها لازم است از فایل VirtualTopologyGeneratorVmTypesSFC استفاده کنیم. این فایل در پوشه‌ی

cloudsimsdn\src\main\java\org\cloudbus\cloudsim\sdn\example\topogenerators

قرار دارد. در این پروژه، مطابق مقاله لازم است که دو توپولوژی متفاوت برای VMها بسازیم. یک توپولوژی با حداکثر منابع و بدون مقیاس‌پذیری برای حالت NoScale-Max و یک توپولوژی با حداقل منابع برای باقی حالت‌ها. برای این که بتوانیم راحت‌تر این توپولوژی‌ها را مطابق با تنظیمات مطلوب مقاله بسازیم، تغییراتی در فایل مربوطه دادیم که در اینجا به بیان تغییرات مهم آن می‌پردازیم. باقی تغییرات جزئی مانند اضافه‌کردن امکان انتخاب بین حالت‌های min و max را می‌توانید با مقایسه‌ی بین فایل اولیه و فایل ما ببینید.

مهم‌ترین تغییر این فایل این است که در مقاله برای حالت حداکثر از ۲۰۰۰۰۰۰ واحد پهنای باند استفاده کرده است و برای باقی حالت‌ها از ۵۰۰۰۰۰ واحد. این تغییر با تکه‌کد زیر اعمال شده است:

```
final ArrayList<Long> linkBW = new ArrayList<>();
if (isMinMode) {
   linkBW.add(500000L);
}
else {
   linkBW.add(2000000L);
}
```

نتیجه‌ی اجرای این کد برای حالت حداکثر و حداقل فایل‌های sfc.virtual.max.json و sfc.virtual.min.json خواهد بود که همانطور که گفتیم، فایل اول برای حالت NoScale-Max استفاده می‌شود و فایل دوم برای باقی حالت‌ها. این فایل‌ها را مانند توپولوژی فیزیکی به پوشه‌ی

cloudsimsdn\project-acn\resources

منتقل می‌کنیم.

ایجاد Workloadها

برای ایجاد Workloadهای این شبیه‌سازی، از فایل‌های ریپازیتوری https://github.com/Cloudslab/sfcwikiworkload استفاده می‌کنیم. نویسندگان مقاله نیز از همین داده‌ها برای ایجاد Workloadها استفاده کرده‌اند. در این ریپازیتوری اسکریپتی برای ایجاد Workloadها از روی داده‌های ویکی‌پدیا وجود دارد.

ما برای راحت‌تر ساختن Workloadها و مطابقت آن‌ها با مقاله، کمی اسکریپت این ریپازیتوری را بهبود دادیم. یکی از تغییرات اصلی ما در این فایل، ست‌کردن مقادیر درست برای اینستنس‌های وب، دیتابیس اپلیکیشن بود:

```
webNum=8,
appNum=24,
dbNum=2,
```

همچنین با توجه به توضیحات مقاله، باید داده‌ها را برای ۲۴ ساعت ایجاد کنیم. به همین دلیل، پارامتر endTime در تابع main_history را روی ۸۶۴۰۰ که معادل ۲۴ ساعت می‌شود ست می‌کنیم:

```
main_history<-function(lang="de", startTime=0, endTime=86400)
```

فایل‌های ما برای این اسکریپت در پوشه‌ی

cloudsimsdn\project-acn\resources\workloads\generator

قرار داده شده است.

اسکریپت به زبان R نوشته شده است. به همین دلیل ابتدا باید R را نصب کنیم. فایل نصبی R برای ویندوز از آدرس https://cloud.r-project.org/bin/windows/base/R-4.2.2-win.exe قابل دانلود است. پس از نصب، باید اسکریپت را در کنسول R اجرا کنیم. سپس تابع main_history() آن را صدا می‌زنیم تا شروع به ایجاد Workloadها بکند.

با توجه به توپولوژی ما، در نهایت ۲۴ فایل بدست می‌آید که از ۰ تا ۲۳ شماره‌گذاری شده‌اند. این فایل‌ها Workloadهای ما خواهند بود. به طور مثال، فایل‌های با اسم‌های زیر ایجاد می‌شوند:

0_0_workload_wiki.csv

0_1_workload_wiki.csv

...

0_23_workload_wiki.csv

فایل‌های حاصل را در پوشه‌ی

cloudsimsdn\project-acn\resources\workloads

قرار می‌دهیم.

آماده‌سازی فایل اصلی اجرای شبیه‌سازی

اولین مرحله برای راه اندازی آزمایش انجام شده در این مقاله پیاده‌سازی شرایط محیطی مفروض در مقاله است. با استفاده از ژنراتورهای معرفی شده در بخش قبل مفروضات محیطی آزمایش را به دست آوردیم در ادامه با ساخت یک آزمایش جدید در محیط CloudSim شروع به کار می‌کنیم. ابتدا فایل ACN.java را در مسیر

\CloudsimSDN\src\main\java\org\cloudbus\cloudsim\sdn\example

می‌سازیم. از میان مثال‌های موجود در نسخه اصلی Cloudsim مثال StartExperimentSFC نزدیک‌ترین شرایط به آزمایش این مقاله را دارد پس ما در ادامه نیز سعی می‌کنیم با استفاده از این مثال بیشتر با قابلیت‌های CloudSim آشنا شویم.

با بررسی مثال StartExperimentSFC متوجه شدیم که به صورت اولیه برای فید کردن ورکلودها دو حالت پیش‌بینی شده است: حالت تک فایلی و حالت چند فایلی. از آنجایی که خروجی ژنراتور ورکلود ما در بخش قبل ۲۴ عدد فایل شده بود پس باید از این حالت چند فایلی استفاده می‌کردیم اما به علت عدم همخوانی نوع خروجی ژنراتور با حالت پیش فرض شبیه‌ساز در این بخش تغییراتی دادیم تا فایل‌ها را به صورت درست دریافت و پردازش کند:

```
if(isInteger(args[n])) {
   // args: <startIndex1> <endIndex1> <filename_suffix1> ...
   int i = n;
   while(i < args.length) {
      Integer startNum = Integer.parseInt(args[i++]);
      Integer endNum = Integer.parseInt(args[i++]);
      String filenameSuffix = args[i++];
      System.out.println(startNum+" ** " +endNum+" ** " +filenameSuffix);
      List<String> names = createGroupWorkloads(startNum, endNum,
filenameSuffix);
      workloads.addAll(names);
   }
```

ابتدا در بخش بالا اضافه کردیم که کاربر در زمان اجرا شماره فایل اول، شماره فایل آخر و پسوند مشترک این فایل‌ها را زمان اجرا به صورت آرگومان‌های جدا به شبیه‌ساز بدهد.

سپس در تابع createGroupWorkloads قابلیت خواندن فایل‌ها را با تغییر ساختار نام‌ها اضافه می‌کنیم.

```
private static List<String> createGroupWorkloads(int start, int end, String
filename_suffix_group) {
   List<String> filenameList = new ArrayList<String>();

   for(int set=start; set<=end; set++) {
      String filename = "resources/workloads/0_" +set +"_" +
filename_suffix_group;
      filenameList.add(filename);
   }
   return filenameList;
}
```

در ادامه باید شرایط آزمایش مربوط به مانیتورینگ و شرایط SLA را برای آزمایش خود مقداردهی کنیم.

با بررسی فایل‌های شبیه‌ساز CloudSim به فایل Configuration.java در آدرس:

CloudsimSDN\src\main\java\org\cloudbus\cloudsim\sdn

با بررسی این فایل متوجه می‌شویم که متغیرهای محیطی مورد نیاز برای آزمایش توسط شبیه‌ساز پیش‌بینی شده‌اند و تنها نیاز به مقداردهی اولیه طبق مقاله دارند، پس در ابتدای آزمایش خود مقادیر لازم را به صورت زیر مقدار دهی می‌کنیم.

```
// Initialize
Configuration.monitoringTimeInterval = 1; // 1 minute
Configuration.TIME_OUT = 10; // 10 seconds SLA
Configuration.workingDirectory = "project-acn/";
```

اما هدف اصلی این آزمایش بررسی روش‌های مختلف مقیاس‌پذیری و تاثیر آنها در کارایی سیستم دارد.

ابتدا سیاست شبیه‌ساز را برای حالت‌هایی که مقیاس‌پذیری داریم تعیین می‌کنیم:

از دیگر آرگومان‌های ورودی که برای این آزمایش در نظر گرفته ایم به ۲ مورد زیر اشاره می‌کنیم:

- میزان اولیه منابع: با انتخاب یکی از دو حالت min و یا max منابع را به صورت اولیه میتوانیم با حداقل یا حداکثر توانایی مقداردهی کنیم.
- مقیاس‌پذیری: آیا در این سناریو منابع مقیاس پذیر هستند یا خیر؟ این ورودی که تنها مقادیر ۰ یا ۱ می‌پذیرد تعیین کننده استراتژی ما در قبال مقیاس‌پذیری منابع است.
- منابع مورد نظر: این ورودی یکی از ۳ حالت MFF، MFFBW و MFFCPU را می‌تواند داشته باشد. در حالت اول هر دو منبع پهنای‌باند و پردازنده همزمان باهم کنترل می‌شوند. در حالت دوم تنها پهنای‌باند کنترل می‌شود و در حالت سوم تنها پردازنده.

با استفاده این ورودی‌ها می‌توانیم پنج سناریوی معرفی شده در مقاله را تعریف و مقایسه کنیم. تفاوت این سناریوها در میزان منابع موجود و استراتژی ما در کنترل آن‌ها است در ادامه جدولی حاوی مقدار حداکثری و

حداقلی این منابع از مقاله ارائه داده‌ایم و سپس این سناریوها را به صورت خلاصه توضیح داده و ورودی‌های لازم برای هرکدام را نشان داده‌ایم:

Table 4
Initial SF resource allocations in each policy.

SF Type	ElasticSFC NoScale-Min			NoScale-Max		
	CPU (Cores*MIPS)	Number of VMs	Bandwidth (MB/s)	CPU (Cores*MIPS)	Number of VMs	Bandwidth (MB/s)
Firewall (FW)	8*10,000	1	500,000	16*10,000	3	2,000,000
Load balancer (LB1)	2*10,000	1	500,000	10*10,000	1	2,000,000
Load balancer (LB2)	2*10,000	1	500,000	10*10,000	1	2,000,000
IDS	6*10,000	1	500,000	12*10,000	3	2,000,000

۱. حداقل منابع: در این سناریو پهنای‌باند و پردازنده در حالت حداقل به صورت پیش‌فرض قرار دارند و هیچ یک از دو منبع در این سناریو مقیاس‌پذیر نیستند و در طول آزمایش مقداری ثابت دارند.
برای اجرای این سناریو باید آزمایش را با آرگومان‌های زیر اجرا کنید:

```
MFF min 0 0 23 workload_wiki.csv
```

۲. حداکثر منابع: در این سناریو پهنای‌باند و پردازنده در حالت حداکثر به صورت پیش‌فرض قرار دارند و هیچ یک از این دو منبع مقیاس‌پذیر نیستند.
برای اجرای این سناریو باید آزمایش را با آرگومان‌های زیر اجرا کنید:

```
MFF max 0 0 23 workload_wiki.csv
```

۳. مقیاس‌پذیری پهنای‌باند: در این سناریو تنها مقدار پهنای باند قابل مقیاس‌پذیری است. مقدار اولیه آن حداقل است و می‌تواند تا مقدار حداکثری به صورت افقی و عمود افزایش یابد. مقدار پردازنده حداقل و ثابت است.
برای اجرای این سناریو باید آزمایش را با آرگومان‌های زیر اجرا کنید:

```
MFFBW min 1 0 23 workload_wiki.csv
```

۴. مقیاس‌پذیری پردازنده: در این سناریو تنها مقدار پردازنده قابل مقیاس‌پذیری است. مقدار اولیه آن حداقل است و می‌تواند تا مقدار حداکثری به صورت افقی و عمود افزایش یابد. مقدار پردازنده حداقل و ثابت است.
برای اجرای این سناریو باید آزمایش را با آرگومان‌های زیر اجرا کنید:

```
MFFCPU min 1 0 23 workload_wiki.csv
```

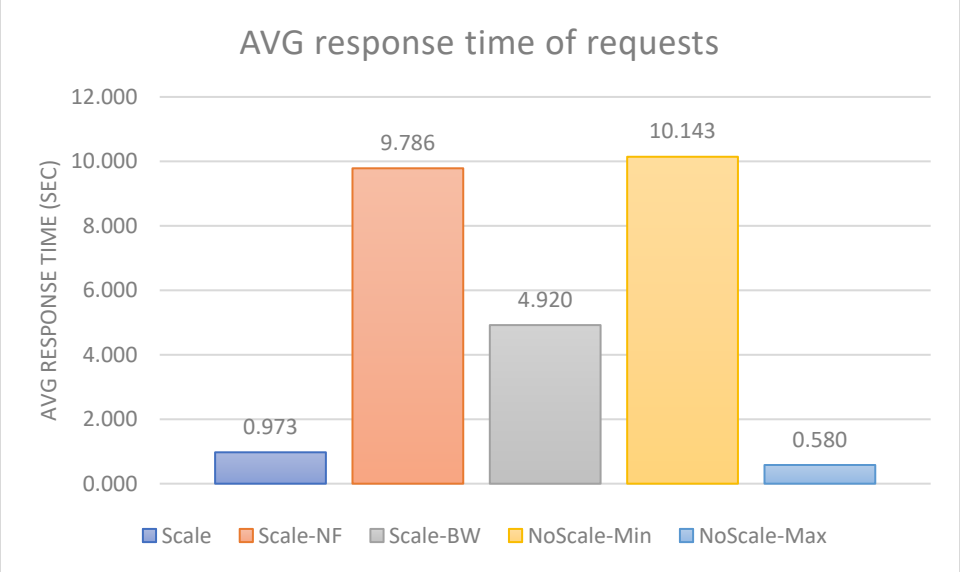
۵. مقیاس‌پذیری کامل: در این سناریو که در اصل روش پیشنهادی نویسندگان مقاله است هر دو منبع میتوانند بر اساس نیاز افزایش یابند یا در صورت عدم نیاز با حفظ میزان کارایی کاهش یابند. در این سناریو هر دو منبع مقدار اولیه حداقلی دارند.

MFF min 1 0 23 workload_wiki.csv

نتایج

هدف ما در این بخش از پروژه بازسازی نتایج کسب شده در مقاله [۱] است. به صورت دقیق‌تر هدف ما مقایسه کارایی سیستم تحت پنج استراتژی مدیریت منابع معرفی شده در بخش قبل است.

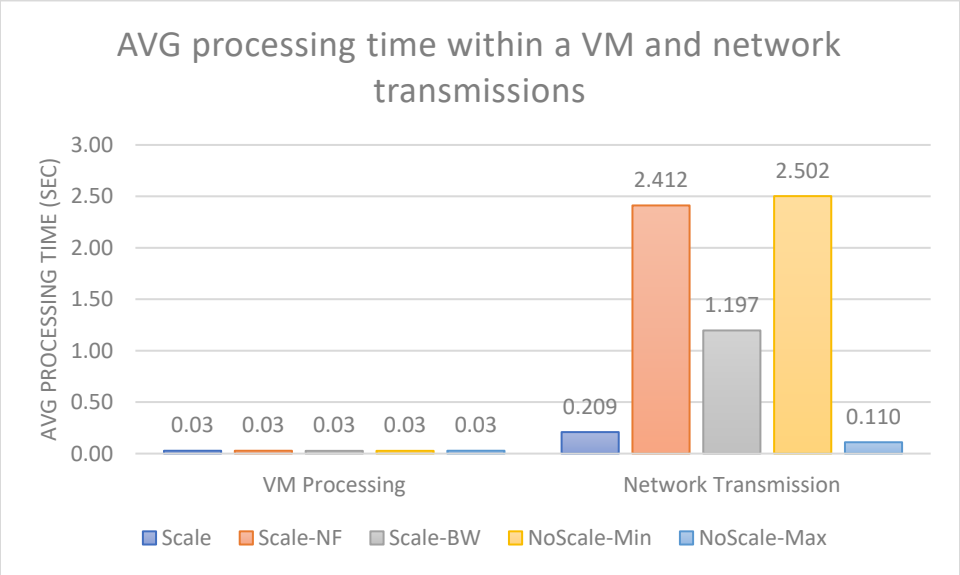
برای ساخت نمودار 6(B) و 6(C) از مقاله ۵ سناریو تعریف شده در بخش قبل را اجرا کردیم و خروجی آن‌ها را در زیر نمایش داده‌ایم.



شکل ۳-۰.میانگین زمان پاسخگویی سیستم به درخواست‌ها

همانطور که در شکل بالا مشاهده می‌کنید مطابق انتظارمان به ترتیب NoScale-Max کمترین و NoScale-Min بیشترین زمان پاسخ دهی را دارند. در بین ۳ سناریو دیگر نکته حائز اهمیت تفاوت بین Scale-NF و Scale-BW است که با نمودار حاصل در مقاله نیز تفاوت‌هایی دارد. در حقیقت در مقاله این دو روش مدیریت خروجی نزدیک به یکدیگر دارند اما در خروجی ما مدیریت پهنای باند زمان پاسخ‌گویی بسیار بهتری از مقیاس‌پذیری پردازنده دارد. برداشت ما از این نمودارها این است که عملا پهنای باند منبع محدودتری در این سناریو است و به همین علت مقیاس پذیری آن تاثیر بیشتری در بهبود کارایی سیستم دارد. در نهایت نتیجه

اصلی مقاله را در سناریوی Scale می‌بینیم که با خروجی‌های مقاله هم‌خوانی دارد و نشان می‌دهد که با مقیاس‌پذیری همزمان هردو منبع می‌توانیم پاسخ‌دهی سیستم را بسیار بهبود دهیم علیرغم اینکه نسبت به سناریو رغم اینکه نسبت به سناریو NoScale-Max منابع بسیار کمتری استفاده می‌کنیم.



شکل ۴-۰ میانگین زمان پردازش و انتقال درخواست‌ها در سیستم

در ادامه به بازسازی شکل 6(B) و 6(C) از مقاله می‌پردازیم. همانطور که در شکل بالا می‌بینید. خروجی‌های ما برای میانگین زمان پردازش با خروجی مقاله همخوانی دارد اما در بخش میانگین زمان انتقال اطلاعات مشاهده می‌کنیم که میانگین زمان انتقال اطلاعات در سناریو Scale-NF نزدیک به سناریو NoScale-Min است. این نتیجه با خروجی مقاله تفاوت دارد اما از نظر منطقی درست به نظر می‌رسد زیرا در سناریوی Scale-NF ما به صورت پیش فرض میزان پهنای باند را حداقل فرض کردیم پس میانگین زمان انتقال باید نزدیک به سناریو NoScale-Min باشد.

بخش ۲: بررسی تأثیر شرایط مختلف

در این بخش تأثیر ایجاد تغییرات در شرایط آزمایش را بررسی می‌کنیم. ما این تغییرات را در آزمایش‌های Scale، Scale-NF و Scale-BW بررسی کرده‌ایم. علت انتخاب این ۳ حالت به شرح زیر است:

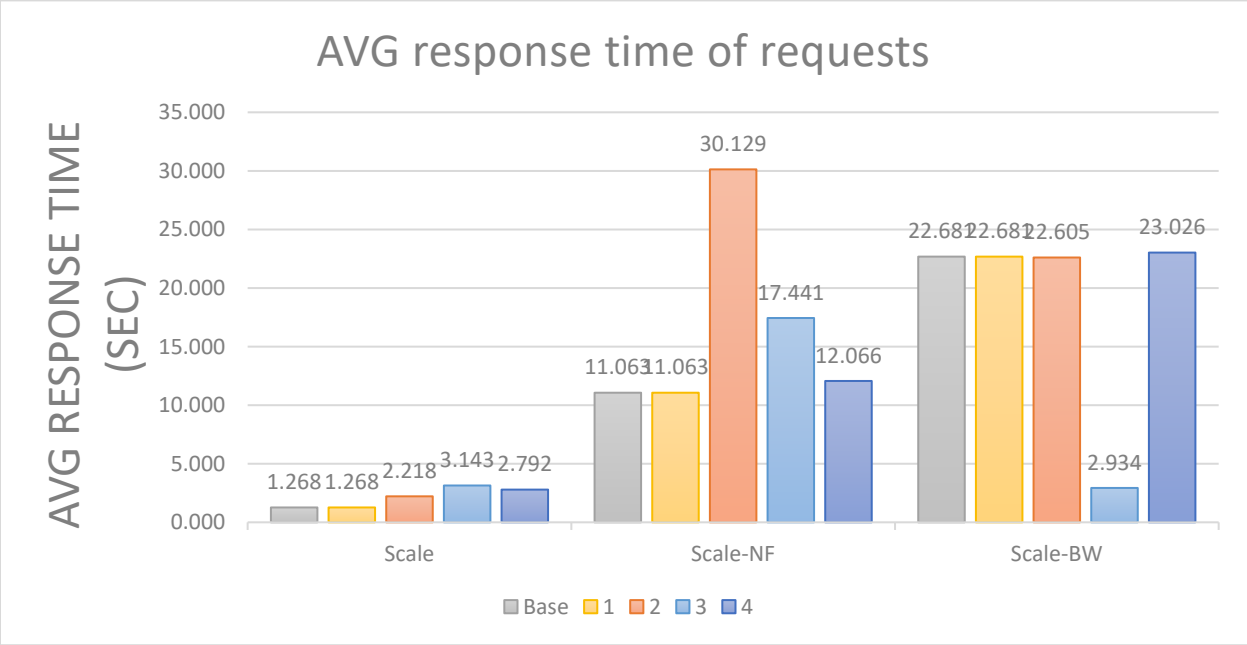
۱. تمرکز اصلی این آزمایش‌ها بر روی عملکرد الگوریتم‌های اسکیلینگ و بررسی تغییر رفتارشان در مواجهه با شرایط متفاوت است.
۲. تأثیر تغییرات بر روی حالت‌های NoScale-Min و NoScale-Max واضح است و بررسی آن‌ها خیلی بار دانشی ندارد.

۳. اجرای هر آزمایش به طور کامل بالای ۶ ساعت در کامپیوتر قوی به طول می‌انجامد. حتی نسخه‌های ۵۰۰۰ ثانیه‌ای (بیش از ۱ ساعت) آزمایش هم حداقل ۳۰ دقیقه زمان می‌گیرد.

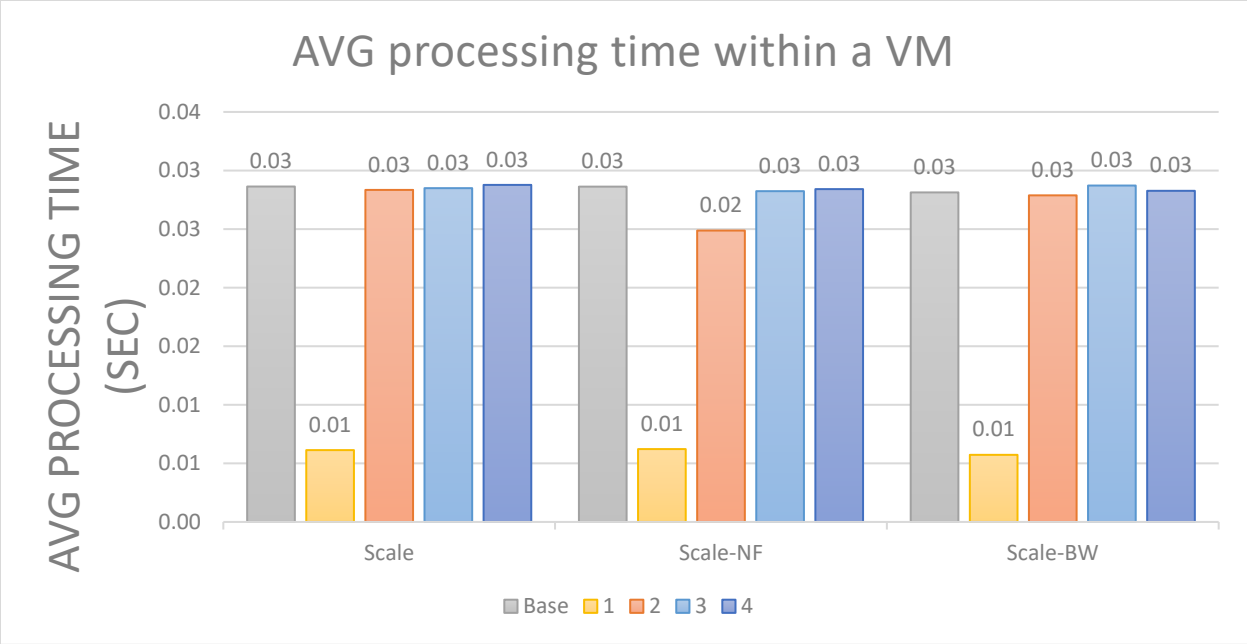
همچنین در این بخش، به خاطر دلیل سوم که بالاتر ذکر کردیم، آزمایش‌ها را بر روی داده‌های کوچک‌تر ۵۰۰۰ ثانیه‌ای اجرا کردیم. از آنجایی که هدف، مقایسه‌ی تفاوت عملکرد است، مشکلی در صحت آزمایش ما ایجاد نمی‌کند.

با توجه به توضیحاتی که دادیم، آزمایش را ابتدا در حالت عادی و بدون هیچ تغییری اجرا کردیم و نتیجه را Base نام نهادیم. سپس برای هر آزمایش، تغییرات مربوط را اعمال کردیم و تمام حالات را اجرا گرفتیم. ما آزمایش‌های شماره‌ی ۱، ۲، ۳ و ۴ را برگزیدیم.

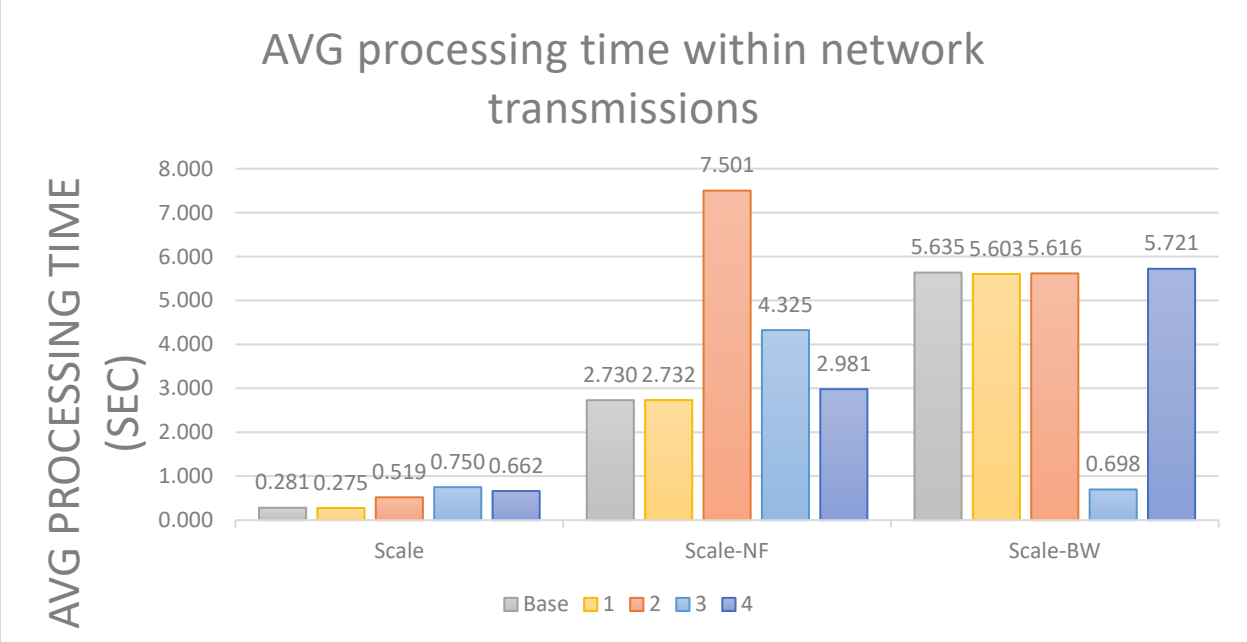
نتیجه‌ی آزمایش‌ها در مقایسه با حالت Base در جداول زیر به تصویر کشیده شده است که در ادامه به تحلیل آن‌ها می‌پردازیم:



شکل ۱-۰مقایسه میانگین زمان پاسخ‌دهی در چهار آزمایش



شکل ۲-۰مقایسه میانگین زمان پردازش در چهار آزمایش



شکل ۳-۰مقایسه میانگین زمان انتقال درخواست‌ها درون شبکه بین آزمایش‌ها

تحلیل آزمایش ۱

در آزمایش ۱ خواسته شده بود که تأثیر تغییر منابع فیزیکی میزبان‌ها را بررسی کنیم. ما در این آزمایش، بار پردازشی درخواست‌ها را کم کردیم. این کار را با ۰.۱ کردن متغیر زیر انجام دادیم:

```
Configuration.CPU_SIZE_MULTIPLY = 0.1;
```

مقدار این متغیر در حالت عادی ۱ است.

همانطور که در نمودارها قابل مشاهده است، زمان لازم برای پردازش درخواست‌ها ۶۶٪ کاهش یافته. ولی زمان کلی درخواست‌ها خیلی تغییر نکرده است. این امر همانطور که از نمودار سوم مشخص است، علتش این است که گلوگاه در شرایط آزمایشی ما پهنای باند است و قدرت پردازشی نیست.

تحلیل آزمایش ۲

در آزمایش ۲ خواسته شده بود که تأثیر تغییر منابع فیزیکی در شبکه را بررسی کنیم. ما در این آزمایش، اندازه‌ی پکت‌ها را ۱۰ برابر کردیم. این کار با تغییر متغیر زیر انجام شد:

```
Configuration.NETWORK_PACKET_SIZE_MULTIPLY = 10;
```

در نمودارها مشهود است که آزمایش ۲ خیلی روی Scale و Scale-BW تأثیر نگذاشته است که طبیعی است؛ زیرا هر دوی این حالت‌ها قابلیت اسکیل‌کردن منابع شبکه را دارند. هرچند به خاطر اسکیل VMها در حالت Scale، با افزایش تعداد VMها و در نتیجه بار شبکه، کمی حالت Scale کندتر شده است ولی با این حال باز هم تأخیر زمان درخواست‌ها به شدت کم نگه داشته شده‌اند. این آزمایش بیشترین تأثیر را روی Scale-NF داشته است که با توجه به این که در این حالت نمی‌توان شبکه را اسکیل کرد، تأخیرها به شدت افزایش یافته و افزایش تعداد VMها هم مزید بر علت شده است!

تحلیل آزمایش ۳

در آزمایش ۳ خواسته شده بود که تأخیر تغییر طول عمر درخواست‌ها را بررسی کنیم. طول عمر درخواست‌ها در حقیقت می‌تواند با تغییر المان MIPS Per Operations تغییر پیدا کند. ما یک توپولوژی جدید به نام sfc.virtual.min.experiment3.json ساختیم که از sfc.virtual.min.json ساخته شده است. در این توپولوژی جدید، المان mipoper در نودهایی که این المان را دارند را ۱۰ برابر کردیم. همچنین در آزمایش‌ها هم از همین توپولوژی جدید استفاده کردیم.

همانطور که از نمودارها مشهود است، به خاطر طول عمر بیشتر درخواست‌ها و در نتیجه ازدحام بیشتر، تأخیر بالاتر رفته است. علت به شدت پایین بودن تأخیر در Scale-BW این است که رسماً به علت کمبود منابع، اغلب درخواست‌ها Timeout می‌خورند و باعث می‌شود که انقدر میانگین پاسخ‌ها کم حساب شود.

تحلیل آزمایش ۴

در آزمایش ۴ خواسته شده بود که تأثیر میزان تأخیر میان زمان تغییرات شبکه و زمان متوجه‌شدن مدیر سیستم را بررسی کنیم. این تأخیر در حقیقت معادل زمان‌های بین هر عمل مونیتورینگ می‌شود. ما بازه‌های بین مانیتورینگ‌ها را با استفاده از متغییر زیر ۱۰۰ برابر کردیم:

```
Configuration.monitoringTimeInterval = 100;
```

در نمودارها مشهود است که این تأخیر بیشتر در زمان مانیتورینگ باعث ایجاد کندی در سرعت عمل سیستم در قبال افزایش بار شده است و شاهد افزایش تأخیر پاسخ‌دهی هستیم. این تغییر در حالت Scale که مقدار خیلی کمی دارد، واضح‌تر است و بالای ۱۰۰ درصد تأخیر پیام‌هایمان بیشتر شده است.

جمع‌بندی

مقاله تحقیقاتی ElasticSFC، یک چارچوب مقیاس‌پذیر خودکار برای زنجیره خدمات شبکه‌ای (SFC) در ابرهای مبتنی بر مجازی‌سازی توابع شبکه (NFV) ارائه می‌کند. هدف ElasticSFC افزایش یا کاهش تعداد توابع شبکه مجازی (VNF) بر اساس بار ترافیک به منظور حفظ کیفیت خدمات (QoS) و در عین حال کارآمد نگه داشتن استفاده از منابع است.

ElasticSFC از چندین معیار برای نظارت بر بار ترافیک و استفاده از منابع VNF ها استفاده می کند. این معیارها شامل استفاده از CPU و حافظه، نرخ پردازش بسته‌ها و استفاده از پهنای باند شبکه است. با تجزیه و تحلیل این معیارها، ElasticSFC می تواند تعداد بهینه نمونه های VNF مورد نیاز برای حفظ تضمین های QoS مورد نظر را تعیین کند.

در این پژوهش ما سعی کردیم تا یافته‌های این مقاله را بازسازی و بررسی کنیم با استفاده از ابزار شبیه‌سازی شبکه Cloudsimsdn توانستیم محیطی مشابه شبکه فرضی در مقاله ایجاد کنیم و همچنین سناریو کاربری مشابه با آنچه در مقاله آزمایش شده بود ایجاد کنیم. در ادامه با افزودن پارامترهای سیستمی و اجرای این آزمایش روی ۵ سناریوی فرضی مقاله توانستیم به نتایجی مشابه با مقاله دست یابیم. یافته‌های ما از این بخش از پژوهش اهمیت مقیاس‌پذیری سیستم در راستای افزایش کارایی می‌باشد. در حالتی که هردو مقیاس‌پذیری

پهنای باند و قدرت پردازش را داشتیم کارایی سیستم نزدیک به حالت حداکثر منابع و میزان منابع مصرف شده نزدیک به حالت حداقل منابع بود که نمایش دهنده میزان کارایی این الگوریتم در حفظ کارایی سیستم علیرغم تغییرات پیش‌بینی ناپذیر ترافیک سیستم می‌باشد.

در بخش دوم از پژوهش سعی کردیم اثر تغییرات در شرایط محیطی فضای آزمایش روی پارامترهای کارایی سیستم را بررسی کنیم. از بین آزمایش‌های موجود در تعریف پروژه آزمایش‌های ۱،۲،۳و۴ را بررسی کردیم. با بررسی آزمایش اول مشاهده‌ای که در بخش یک هم دیده بودیم تکرار شد. کاهش چشمگیر میزان قدرت پردازشی تاثیر کمی روی کارایی سیستم داشت و این نمایش می‌داد که از بین دو منبع موجود پهنای باند منبع محدود کننده کارایی سیستم می‌باشد.

در آزمایش دوم با افزایش بار ترافیک شبکه اهمیت الگوریتم‌های مقیاس پذیری را بهتر متوجه می‌شویم زیرا در این سناریو حالت‌هایی که پهنای باند را تغییر نمی‌دهند شدیدا دچار مشکل می‌شوند.

در آزمایش سوم این بار بار پردازشی درخواست‌ها را افزایش دادیم و مانند آزمایش دوم دیدیم که بدون مقیاس‌پذیری درست بسیاری از درخواست از به علت timeout از بین می‌روند.

در آزمایش سوم ارزشمندی و حساسیت نرخ مانیتورینگ را بررسی کردیم. طبق آزمایش انجام شده حتی اگر از بهترین الگوریتم‌های مقیاس‌پذیری استفاده کنیم اما به صورت لحظه‌ای از شرایط سیستمی مطلع نباشیم، کارایی سیستم به شدت پایین می‌آید.

نهایتا تمامی این آزمایش‌ها بدون فضای شبیه سازی‌ای مانند Cloudsim ممکن نبود. با استفاده از چنین ابزارهایی تست و بررسی شبکه‌های پیچیده و سیستم‌های مانیتورینگ پیچیده بسیار راحت تر و با هزینه بسیار کمتر ممکن شده است و از این جهت چنین ابزارهای شبیه‌سازی‌ای برای تحقیق و توسعه بسیار حیاتی می‌باشند.

منابع

[1] Toosi, A. N., Son, J., Chi, Q., & Buyya, R. (2019). ElasticSFC: Auto-scaling techniques for elastic service function chaining in network functions virtualization-based clouds. *Journal of Systems and Software*, *152*, 108-119.