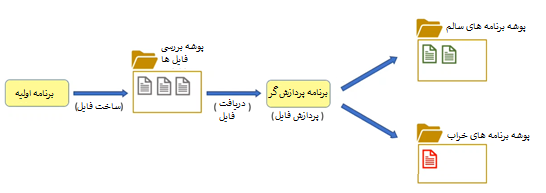
گزارش چهارم

مقدمه

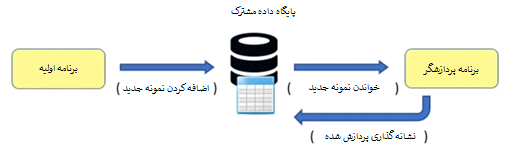
یک سیستم کامل، از چند برنامه تشکیل‌شده که وظایف متفاوتی را انجام می­دهند، در اغلب مواقع این برنامه‌ها نیاز دارند با یکدیگر تعامل داشته باشند و باهم ارتباط برقرار کنند. به ایجاد کردن ارتباط میان برنامه‌های متفاوت یکپارچه‌سازی[[1]](#footnote-1) می­گویند. شیوه‌های متفاوتی از یکپارچه­سازی وجود دارد.

مدل اول یکپارچه­سازی مبتنی بر فایل است. در این مدل، برنامه اولیه فایلی که باید پردازش شود را ایجاد می­کند و در پوشه‌ای مشخص قرار می­دهد، سپس برنامه‌ای دیگر فایل‌هایی که در این پوشه قرارگرفته‌اند را بررسی می­کند. برای مثال برنامه دوم و فایل­های خراب را از سالم جدا کرده و در پوشه‌هایی متفاوت قرار می­دهد. قابل‌مشاهده است که این دو برنامه می­توانند مستقل از هم کار کنند و حتی می­توانند با زبان‌های برنامه‌نویسی متفاوتی ایجادشده باشند. در شکل ‏4–1 می­توان مثال فوق را مشاهده کرد.



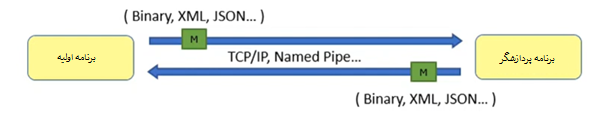
شکل ‏4–1-مثالی از یکپارچه‌سازی مبتنی بر فایل

مدل دوم یکپارچه­سازی­هایی مبتنی بر پایگاه­داده مشترک یا توزیع‌شده است. در این مدل ابتدا یکی از برنامه­ها اطلاعاتی را در پایگاه­داده تغییر می­دهد. سپس برنامه دیگری آن را برداشته و پردازش می­کند و مجدداً به پایگاه­داده برمی­گرداند و به آن برچسب پردازش‌شده می­زند. به‌طور مشابه، این دو برنامه می­توانند مستقل از یکدیگر فعالیت کنند. در شکل 4–2 می­توان مدل فوق را بررسی نمود.



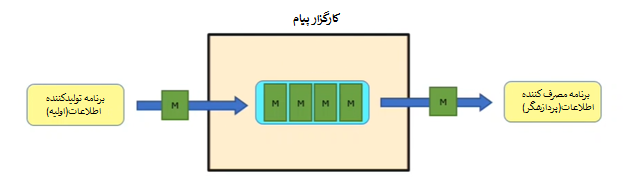
شکل ‏4–2-مثالی از یکپارچه­سازی مبتنی بر پایگاه­داده

مدل سوم یکپارچه­سازی­هایی مبتنی بر ارتباط مستقیم است، به‌نحوی‌که برنامه‌ها به‌صورت مستقیم با یکدیگر ارتباط tcp/ip برقرار کرده و بعد از برقراری ارتباط شروع به ارسال پیام به یکدیگر می­کنند. فرمت پیام‌ها می­تواند به گونه‌های متفاوتی ازجمله باینری یا مبتنی برنوشته مانند, xml, json ... باشد. در شکل ‏4–3 می­توان نمونه‌ای از این ارتباط را مشاهده کرد.



شکل ‏4–3-نمونه از یکپارچه­سازی بر اساس ارتباط مستقیم میان برنامه‌ها

مدل چهارم یکپارچه­سازی بر پایه پیام دادن نامتقارن با کمک یک کارگزار پیام[[2]](#footnote-2) است. برنامه اولیه یا تولیدکننده اطلاعات در یک‌سو با هر فرمتی که بخواهد می­تواند، پیام ارسال کند ولی این باریک برنامه میانی پیام‌ها را دریافت می­کند (که اغلب به آن message broker یا message bus می­گویند) پیام‌ها را در یک لیست قرار داده که به آن Queue یا صف می­گویند و آن‌ها را به هدف، مقصد یا مصرف‌کننده منتقل می­کند. در شکل ‏4–4 می­توان نمونه‌ای از این شیوه انتقال اطلاعات را مشاهده کرد. این روش، یک روش محبوب است که در سیستم­های متفاوتی می­تواند مورداستفاده قرار بگیرد.



شکل ‏4–4-نمونه‌ای از یکپارچه­سازی با کمک یک کارگزار پیام

این شیوه از یکپارچه‌سازی مزایای بسیاری دارد که به شرح زیر است:

* جدا بودن برنامه تولیدکننده از برنامه مصرف‌کننده:
  + برنامه تولیدکننده اطلاعات و برنامه مصرف‌کننده نیازی ندارد یکدیگر را بشناسند.
  + آدرس و تکنولوژی‌های مورداستفاده یکدیگر را نمی­دانند.
  + تنها چیزی که نیاز است نسبت به هم آگاه باشند، فرمت اطلاعات ارسالی و دریافتی است.
* ایجاد محیطی مطمئن، قابل‌اعتماد برای انجام پردازش‌ها و ارتباط میان برنامه­ها:
  + برنامه‌های تولیدکننده اطلاعات می­تواند به ارسال اطلاعات بپردازد، درحالی‌که هیچ برنامه‌ای در طرف دیگر، اطلاعات را مورداستفاده قرار نمی­دهد، به این معنا که اطلاعات در برنامه میانی به‌صورت موقت ذخیره می­شوند و در ادامه هرگاه که برنامه مصرف‌کننده اطلاعات، فعالیت خود را آغاز کرد، می­تواند از اطلاعات استفاده نماید.
  + درصورتی‌که برنامه مصرف‌کننده در پردازش شکست بخورد، می­تواند درخواست ارسال مجدد داشته باشد.
* فراهم کردن راهی برای پیاده­سازی معماری­هایی افقی در تعداد بالا
  + اگر برنامه مصرف‌کننده اطلاعات به‌تنهایی نتواند همه پیام‌ها را پردازش کند، می­توان چندین مصرف‌کننده اطلاعات را در کنار هم قرارداد تا فرایند سریع­تر انجام شود.

این شیوه بسیار پربازده‌تر از مدل‌های یکپارچه­سازی مبتنی بر پایگاه­داده، است.

جمع بندی:

در این پروژه به دلیل نیاز به انتقال اطلاعات دوربین ها در شبکه، به مدل چهارم یکپارچه سازی نیاز داریم تا بتوانیم تصاویر دوربین ها را در شبکه از یک کامپیوتر به یک کامپیوتر دیگر انتقال دهیم. برای انجام اینکار از نرم افزار rabbitmq استفاده شد.

و در این بخش سعی شد که تنها یک فریم از تصویر را توسط این نرم افزار منتقل نمود. به همین منظور یک برنامه برای ارسال اطلاعات به شکل زیر نوشته شد:

import  pika , pickle

import numpy as np

frame = pickle.load( open( "a.p", "rb" ) )

credentials = pika.PlainCredentials('guest', 'guest')

parameters = pika.ConnectionParameters('localhost',

                                       5672,

                                        '/',

                                        credentials)

channel=pika.BlockingConnection(parameters).channel()

channel.basic\_publish(

            exchange='e.R',

            routing\_key='c.1',

            body=frame.tobytes(),

            properties=pika.BasicProperties(delivery\_mode = 2,)

            )

و در سمت دیگر، یک برنامه برای دریافت اطلاعات از سرور نوشته شد. که در ادامه قابل مشاهده است. پارامتر های بکار رفته را در گزارش های بعدی با جزئیات بیشتر شرح خواهم داد چرا که نیاز به تغییرات زیادی دارد.

import  pika

import numpy as np

import cv2 as cv

import pickle

frame = pickle.load( open( "a.p", "rb" ) )

def Packet\_Handeler\_callback(ch, method, properties, body):

    X=np.frombuffer(body,dtype=np.dtype('uint8'))

    frame=X.reshape(1080, 1920, 3)

    cv.imshow('frame',  frame  )

    cv.waitKey()

    cv.destroyAllWindows()

    ch.basic\_ack(delivery\_tag = method.delivery\_tag)

credentials = pika.PlainCredentials('guest', 'guest')

parameters = pika.ConnectionParameters('localhost',

                                       5672,

                                        '/',

                                        credentials)

channel=pika.BlockingConnection(parameters).channel()

channel.basic\_qos(prefetch\_count=1)

channel.basic\_consume(queue='Cam1',

              on\_message\_callback=Packet\_Handeler\_callback,

              consumer\_tag='1')

print(' [\*] Waiting for messages')

channel.start\_consuming()

1. integration [↑](#footnote-ref-1)
2. Message Broker [↑](#footnote-ref-2)