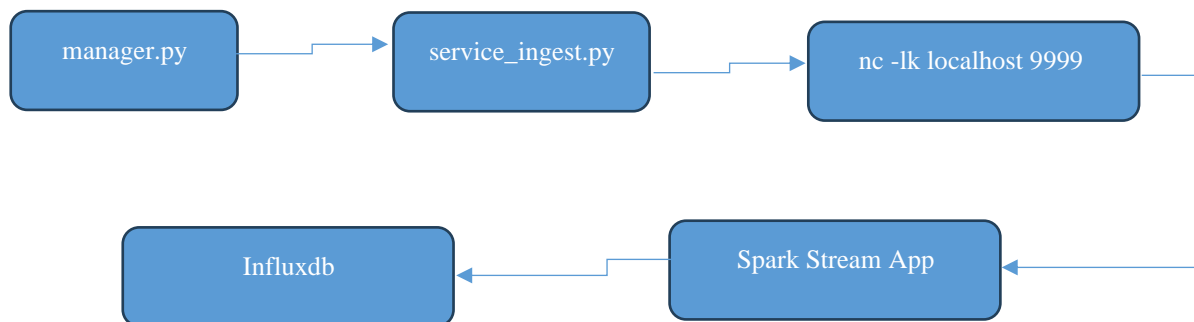


۱. معماری کلی سامانه

جهت پردازش داده‌های جریانی ابتدا از یک کد پایتونی به عنوان واسط استفاده شده است تا اطلاعات ارسالی از طرف manager.sh دریافت و برای کلاستر پردازش جریانی ارسال نماید. جهت پردازش جریانی از Spark استفاده شده است. Spark داده‌های جریانی پردازش نموده و معیارهای EMS، MS و RSI را محاسبه می‌کند و خروجی را به Influxdb جهت بصری‌سازی بلادرنگ ارسال می‌نماید. شماتیک کلی معماری سامانه در شکل زیر آمده است.



۲. service_ingest.py

همانطور که پیام شد این بخش وظیفه دریافت داده‌های ارسالی از manager.py را بر عهده دارد.

```
service_ingest.py > ...
1  from flask import Flask, request, jsonify
2
3  app = Flask(__name__)
4
5  @app.route('/ingest', methods=['POST'])
6  def ingest():
7      try:
8          data = request.json # Assumes the incoming data is JSON
9          print(f"{data}")
10
11         return jsonify(data)
12     except Exception as e:
13         return jsonify({"status": "error", "message": str(e)}), 500
14
15 if __name__ == '__main__':
16     app.run(port=5000, debug=True)
17
```

این برنامه ورودی دریافتی را در قالب json چاپ می‌نماید. نمونه‌ای از خروجی کد بالا در ادامه آمده است.

```
* Restarting with stat
* Debugger is active!
* Debugger PIN: 520-282-815
{'stock symbol': 'GOOGL', 'opening_price': 993.6634706262023, 'closing_price': 998.6334706262023, 'high': 998.9234706262023, 'low': 987.4234706262023, 'volume': 5423, 'timestamp': 1707329024.4773145}
127.0.0.1 - - [07/Feb/2024 21:33:44] "POST /ingest HTTP/1.1" 200 -
{'stock symbol': 'AAPL', 'opening_price': 1007.8752504739059, 'closing_price': 993.825250473906, 'high': 1014.1252504739059, 'low': 980.7352504739059, 'volume': 5204, 'timestamp': 1707329027.3593557}
127.0.0.1 - - [07/Feb/2024 21:33:47] "POST /ingest HTTP/1.1" 200 -
{'stock symbol': 'MSFT', 'opening_price': 980.6458134111325, 'closing_price': 974.5358134111325, 'high': 984.8958134111325, 'low': 973.7258134111325, 'volume': 5644, 'timestamp': 1707329031.6445296}
127.0.0.1 - - [07/Feb/2024 21:33:51] "POST /ingest HTTP/1.1" 200 -
{'stock symbol': 'GOOGL', 'opening_price': 1003.204401903608, 'closing_price': 999.834401903608, 'high': 1004.5144019036079, 'low': 998.954401903608, 'volume': 4122, 'timestamp': 1707329036.3018084}
127.0.0.1 - - [07/Feb/2024 21:33:56] "POST /ingest HTTP/1.1" 200 -
{'stock symbol': 'GOOGL', 'opening_price': 1009.7376087605563, 'closing_price': 1011.7176087605563, 'high': 1014.4176087605564, 'low': 1006.9176087605563, 'volume': 5470, 'timestamp': 1707329039.8218565}
127.0.0.1 - - [07/Feb/2024 21:33:59] "POST /ingest HTTP/1.1" 200 -
{'stock symbol': 'TSLA', 'opening_price': 1011.7481059543564, 'closing_price': 1017.5381059543564, 'high': 1018.0181059543564, 'low': 1011.6081059543565, 'volume': 3995, 'timestamp': 1707329042.249479}
127.0.0.1 - - [07/Feb/2024 21:34:02] "POST /ingest HTTP/1.1" 200 -
{'stock symbol': 'GOOGL', 'opening_price': 992.4211583390339, 'closing_price': 970.741158339034, 'high': 993.0811583390339, 'low': 967.111158339034, 'volume': 4495, 'timestamp': 1707329044.885629}
```

۳. استفاده از دستور netstat

در ادامه با استفاده از دستور زیر داده‌ها را برای Spark ارسال می‌نماییم. به‌طور پیش‌فرض Spark داده‌ها را از روی پورت 9999 می‌خواند.

```
^C(.venv) mjf@k8smaster:~/Desktop/DS_Project$ python3 service ingest.py | nc -lk localhost 9999
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 520-282-815
127.0.0.1 - - [07/Feb/2024 21:36:49] "POST /ingest HTTP/1.1" 200 -
127.0.0.1 - - [07/Feb/2024 21:36:52] "POST /ingest HTTP/1.1" 200 -
127.0.0.1 - - [07/Feb/2024 21:36:57] "POST /ingest HTTP/1.1" 200 -
127.0.0.1 - - [07/Feb/2024 21:36:59] "POST /ingest HTTP/1.1" 200 -
127.0.0.1 - - [07/Feb/2024 21:37:03] "POST /ingest HTTP/1.1" 200 -
127.0.0.1 - - [07/Feb/2024 21:37:07] "POST /ingest HTTP/1.1" 200 -
```

۴. برنامه پردازش جریانی داده‌ها مالی

در این مرحله نرم‌افزاری با استفاده از pyspark توسعه داده شده که داده‌ها را به‌طور مداوم از پورت ۹۹۹۹ می‌خواند و برای پردازش به تابع محاسبه کننده معیارها و استخراج سیگنال ارسال می‌نماید. در بخش main برنامه داده‌ها به‌صورت جریانی توسط Spark خوانده می‌شوند و به ازای هر Batch برای تابع update_dict فراخوانی خواهد شد.

```

145
146 if __name__ == "__main__":
147
148
149     sc = SparkContext("local[2]", "SocketStreamExample")
150     # set batch interval to 1 second (minimum value)
151     ssc = StreamingContext(sc, 1)
152
153     # Create a DStream that will connect to a socket and receive raw binary data
154     samples = ssc.socketTextStream("localhost", 9999)
155
156     #json_data_stream = samples.map(lambda line: json.loads(line))
157     #print(json_data_stream)
158
159     # Update dictionary with new data
160     # samples.foreachRDD(lambda rdd: rdd.foreach(update_dict))
161     samples.foreachRDD(update_dict)
162
163     # Process each sample
164     #samples.foreachRDD(lambda rdd: rdd.foreach(process_sample))
165
166     # Start the streaming computation
167     ssc.start()
168
169     # Keep the program running
170     try:
171         # Sleep for a long time to keep the Spark Streaming context running
172         # You can interrupt the program when you want to stop it
173         ssc.awaitTermination() # Timeout set to 24 hours (in seconds)
174     except KeyboardInterrupt:
175         # Stop the Spark Streaming context if interrupted
176         ssc.stop()
177

```

بخش اصلی کد به همراه گزارش ارسال شده است و در اینجا جهت پرهیز از طولانی شدن از آوردن آن صرفنظر می‌شود. در نهایت خروجی برنامه اسپارک در کنسول اطلاعات زیر را چاپ می‌نماید.

```

File Edit Selection View Go Run Terminal Help
EXPLORER
DS_PROJECT
  .venv
  .vscode
  {} launch.json
  > dpenv
  data_sender.py
  real_time_processing.py
  requirements.txt
  service_ingest_kafka.py
  service_ingest.py
  web_service_data_rec.py

real_time_processing.py > calculate_signals
96     ems_threshold = np.mean(ems)
97     print("")
98     print(ms_threshold)
99     print(ems_threshold)
100    print(rsi_threshold)
101    print("")
102    # Define signals based on thresholds
103    rsi_signal = 'Buy' if rsi < rsi_threshold else 'Sell'
104    ms_signal = 'Buy' if mean_square < ms_threshold else 'Sell'
105    ems_signal = 'Buy' if ems[-1] < ems_threshold else 'Sell'

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

532.3072010766101
502797.388462611
55

Signals for TSLA:
RSI Value: 6.140480887278741
RSI Signal: Buy
Mean Square Value: 494584.39242262807
Mean Square Signal: Sell
Exponential Mean Square Value: 0.09968903538457739
Exponential Mean Square Signal: Buy

535.6793363182557
80183.4447413418
55

Signals for MSFT:
RSI Value: 0.0
RSI Signal: Buy
Mean Square Value: 72165.10026720837
Mean Square Signal: Sell
Exponential Mean Square Value: 1.0103114037409179e-07
Exponential Mean Square Signal: Buy

```