Assignment #1- Data Models and Communication

Avkash Sharma
Computer Engineering Undergraduate 40012077
COEN 424
Concordia University, Montreal
avkaashsharma89@gmail.com

Karthikan Jeyabalan
Computer Engineering Undergraduate 40032932
COEN 424
Concordia University, Montreal
jeyabalan.karthikan@gmail.com

***Abstract*—This document highlights the implementation details of assignment one. It contains section illustrating steps required to run the application, the design of the data model, methods used for data serialization/deserialization, communication between the data model and the serialization methods, libraries and software packages used and screenshots of successful**

Keywords*—gRPC, REST API, Protobuf, Web API, Server, Client

I. Introduction

A client-server program was created to serve a "workload query" whereby a client's "Request For Workload (RFW)" was replied by the server's "Response For Data (RFD)" for each conversation. The program prompts the client to RFW by allowing the user to enter RFW ID, Benchmark Type, Workload Metric, Batch Unit, Batch ID, Batch Size and server handles the request by replying back the RFW ID, the last Batch ID and the samples requested. The data was communicated between the client and server through data serialization/ deserialization. For text based deserialization an ASP.NET CORE 3 API was created that would provide output in the JSON format through an HTTP request. For binary based deserialization gRPC, a protocol buffer based remote procedure call (RPC) framework over HTTP/2 was used. Both the client and the server programs were created using C# programming language. The report elaborate on the technical details adopted to achieve the requirements laid out in the assignment. It provides details on the procedure to run the application, design of the data model, methods used for data serialization/deserialization, the data communication of the application, libraries and software packages used and the screenshots of running our applications with successful results.

II. Running Our Application

To run all of our application, .NET Core 3 is required to be installed. It is multi-platform and works the best on Windows. Ready to execute applications are located under WorkloadAPI\Releases.

A. Text based serialization/deserialization

In order to satisfy the text based serialization/deserialization, we opted to use a .NET CORE API which would act as a server and Postman as the client. The server is hosted on an ec2 instance at this specific address *https://ec2-34-207-128-240.compute-1.amazonaws.com/response* or it can also be run locally.

Server

1. Change directories into "WorkloadAPI\Releases\restAPI"
2. Run the command "dotnet Workload.dll"
3. The application will start and open a terminal with the URL *(http://localhost:5000)*(Figure 4)

Client

1. Install Postman
2. Make a GET request to *http://localhost:5000/response* or *https://ec2-34-207-128-240.compute-1.amazonaws.com/response* with a request body with the following JSON

```
{
    "RfwID": "10",
    "BenchmarkType" :0,
        "WorloadMetric": 0,
        "BatchUnit": 5,
        "BatchId": 3130,
        "BatchSize": 10
    }
```
(Figure 5)

3. This will output the response. (Figure 6)

B. *Binary Based serialization/Deserialization*

For the binary data serialization/deserialization, there are 2 separate standalone windows executable files (.dll) which are both console application. Once the server is running, we may start the client application. The client application will prompt the different inputs needed to pass on the server.

Server

1. Change directories to “WorkloadAPI\Releases\CommunicationServer”
2. Run command : “dotnet CommunicationServer.dll”
3. The application will start and open a terminal with URL (https://localhost:5001) (Figure 7)

Client

1. Change directories to WorkloadAPI\Releases\GrpcClient
2. Run command: “dotnet GrpcGreeterClient.dll”
3. The application will start in a terminal (Figure 8)
4. Application will ask to enter the client URL and port. Type: https://localhost:5001 (Figure 9)
5. Enter the request information as console will ask. (Figure 10)
6. Application will return response (Figure 11 & 12)
7. The application runs in a loop until “quit” is typed.

III. Design of The Data Model

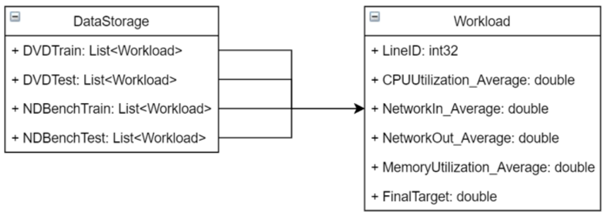

DataStorage
+ DVDTrain: List<Workload>
+ DVDTest: List<Workload>
+ NDBenchTrain: List<Workload>
+ NDBenchTest: List<Workload>
Workload
+ LineID: int32
+ CPUUtilization_Average: double
+ NetworkIn_Average: double
+ NetworkOut_Average: double
+ MemoryUtilization_Average: double
+ FinalTarget: double

Figure 1: Conceptual Data model of Workload and DataStorage entities

Figure 1 illustrates the DataStorage and Workload entities and the relationship between the two. Each attribute in the DataStorage entity contains list of workload object. Workload class has six attributes. LineID of type int32, CPUUtilization_Average of type double, NetworkIn_Average of type double, NetworkOut_Average of type double, MemoryUtilization_Average of type double and FinalTarget of type double. The DataStorage class is used to read data from the given datasets as soon as the application is started. Each element of the dataset is read into the workload lists which are used in the application to request the data from. For instance, during text based deserialization Data.populateList in the Program.cs populates the Workload lists in the DataStorage entity as soon as the application was started.

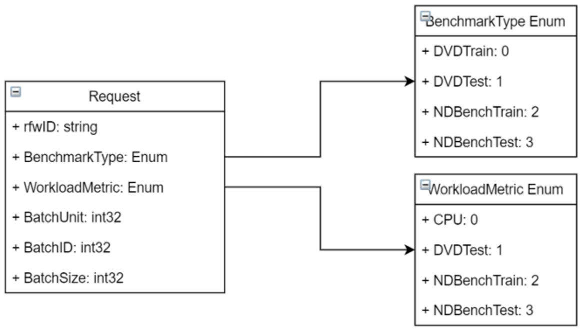


Figure 2: Conceptual Data Model of Client Request

Figure 2 displays the Request entity of our application. It contains six attributes in the form of rfwID of type string, BenchmarkType of type Enum, WorkloadMetric of type Enum, BatchUnit of type int32, BatchID of type int32 and BatchSize of type int32. Two enumeration types in the form of BenchmarkType and WorkloadMetric are used within the request entity. The BenchmarkType enumeration contain the names of the four workload datasets provided to us. User can request data from DVDTraining, DVDTesting, NDBenchTraining or NDBenchTesting. The WorkloadMetric enumeration contain the names of the type of data requested from the dataset. User can request CPUUtilizationAverage, NetworkInAverage, NetworkOutAverage or MemoryUtilizationAverage.

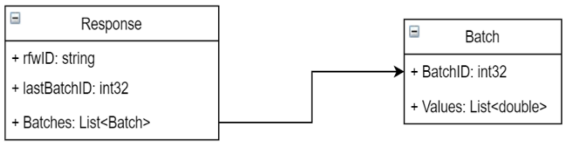


Figure 3: Conceptual Data Model of Server Request

Figure 3 displays the Response entity of our application. It contains three attributes that will be responded back to the client by the server upon client request. The server reply includes the rfwID of type string that the client had used to request the data with, the lastBatchID of type int32 that contains the ID of the last batch returned by the server and Batches of type list of Batch objects. Batch is a class that contains a BatchID attribute of type int32 and Values of type list of double values. Basically, the server sends list of batch objects to the client.

The client server communication for both the text based and binary based methods occur over HTTP. The text based RESTful API communicate via HTTP/1 whereas gRPC by default uses HTTP/2 as its transport protocol.

IV. Methods Used to Serialize/Deserialize Data

Two distinct methods both built on ASP.NET core framework were used to achieve both text based as well as binary serialization/deserialization. JSON was used for text based deserialization and protocol buffer for binary deserialization. gRPC which is a high-performance Remote Procedure Call (RPC) framework uses protocol buffers as the interface definition language for describing both the service interface and the structure of the payload messages by default was integrated to .NET Core [1].

The requested data from the workload dataset was serialized to JSON. The web API was developed using ASP.NET Core. ASP.NET Core provides automatic serialization via the "System.Text.Json" APIs when accepting or returning object payloads. The built-in library "System.Text.Json" serialize the object to JSON and writes the JSON into the body of the response message [2].The HTTP GET endpoint is implemented in the "ResponseController.cs".

In order to achieve binary based serialization/deserialization gRPC framework was hosted on .NET Core. The Microsoft .NET Core framework 3.0 includes tooling and native support for gRPC [3], therefore gRPC was integrated into the .NET Core framework by adding NuGet packages. gRPC allow us to define a service using protocol buffers which is a binary serialization toolset and language. gRPC by default uses Protobuf messages to send and receive data. Protocol Buffers are a method of serializing structured data. It is useful in developing programs to communicate with each other over a wire or for storing data. The method involves an interface description language that describes the structure of some data and a program that generates source code from that description for generating or parsing a stream of bytes that represents the structured data [4]. The protobuf message types are specified in "WorkloadAPI\Code\GrpcGreeterClient\Protos\work.proto" file included in both client and server application. The compiled protobuf message provided simple accessors for each defined field in the "work.proto" as well as methods to serialize/parse the whole structure to/from raw bytes [5].

V. Data Model and Serialization Method Communication

We have created a RESTful API application for text based serialization and a gRPC application for binary based serialization. The web API exchanges JSON data over HTTP/1 whereas the gRPC exchanges binary data over HTTP/2.

In the Web API, a GET request is made to the server endpoint via postman client over HTTP/1. The postman client sends a JSON in the GET request body, the server uses JSON deserialization to convert it to a C# object. And when the server returns an object in its response, it serializes that object into JSON using the built-in library in ASP.NET Core called "System.Text.Json".

In the gRPC application, the client sends a single request (WorkloadRequest) to the server and gets a single response back (WorkloadResponse) via service definition defined in the "work.proto" file. gRPC uses HTTP/2 for client server communication. HTTP/2 breaks down the HTTP protocol communication into an exchange of binary encoded frames. Thus, in our application the request and response occurs as an exchange of binary encoded frames that are mapped according to the messages defined in the "work.proto" file. Thus, the service definition in the proto file serialize/deserialize data as per message definition which represents our data model in protobuf.

VI. Libraries and Software Packages for Data Serializations

A. Libraries for text based serialization

For text based data serialization/deserialization the Web API was created on ASP.NET Core and ASP.NET Core uses a built-in library called "System.Text.Json" to serialize the object to JSON and writes the JSON into the body of the response message [2].

PROS

- Provide high-performance JSON APIs as the library process UTF-8 directly without having to transcode to UTF-16 string instances
- Remove Json.NET dependency from ASP.NET Core
- Can be used with any kind of REST API Client

CONS

- Text based serialization/deserialization is slower than binary based serialization/deserialization.

B. Libraries for Binary based data serialization

gRPC service was used with ASP.NET Core for binary based data serialization/deserialization. gRPC uses protocol buffers (protobuf) as its interface definition language (IDL) by default. Protocol buffer is a binary serialization toolset and language. Following three NuGet Packages were added to our .NET Core application:

1. Google.Protobuf (C# runtime library for Protocol Buffers)
2. Grpc.Net.Client (A gRPC client for .NET Core that builds upon the familiar HTTP Client. The client uses new HTTP/2 functionality in .NET Core) [6]
3. Grpc Tools - gRPC and Protocol Buffer compiler to manage C# projects. Added to a project that contains .proto file to be compiled to code

PROS [7]

- Modern, high-performance, lightweight RPC framework
- Supports client, server, and bi-directional streaming calls
- Reduced network usage with Protobuf binary serialization
- Contract-first API development, using Protocol Buffers by default, allowing for language agnostic implementations

CONS

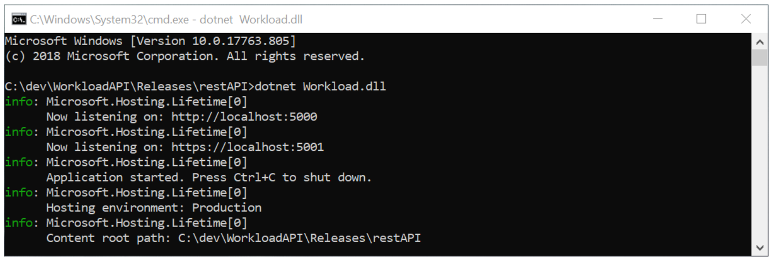
- ASP.NET Core gRPC is not currently supported on Azure App Service or IIS. The HTTP/2 implementation of HTTP.Sys does not support HTTP response trailing headers which gRPC relies on *[8]*.
- ASP.NET Core gRPC app is not fully supported on macOS. Kestrel doesn't support HTTP/2 with TLS on macOS and older Windows versions such as Windows 7. The ASP.NET Core gRPC template and samples use TLS by default *[9]*

VII. Running On Cloud Instance

This assignment was also hosted on an AWS ec2 instance. This instance is running Amazon Linux which comes with .net preinstalled and is a t2.micro type. First, Git and httpd(apache) was installed on the machine. Secondly, the repo was pulled, build and ran on the instance. Only then apache was configured to listen on port 80 and reverse proxied *http://localhost:5000*. This will allow call made to port *http://ec2-34-207-128-240.compute-1.amazonaws.com:80* to reroute to localhost:5000. Figure 13 & 14 represents a sample request to the instance.

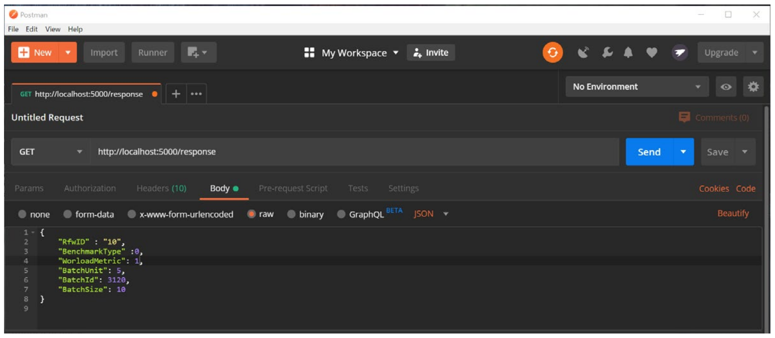
A demonstration video is located at *https://drive.google.com/open?id=17Q9BOOtR7NG3a-BIYbwExoWg5fHk9WYi* . The video might need to be downloaded since it's a 2k resolution and might not play in the browser.

VIII. Screenshots



```
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\dev\WorkloadAPI\Releases\restAPI>dotnet Workload.dll
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\dev\WorkloadAPI\Releases\restAPI
```

Figure 4: Running the Rest API server in console



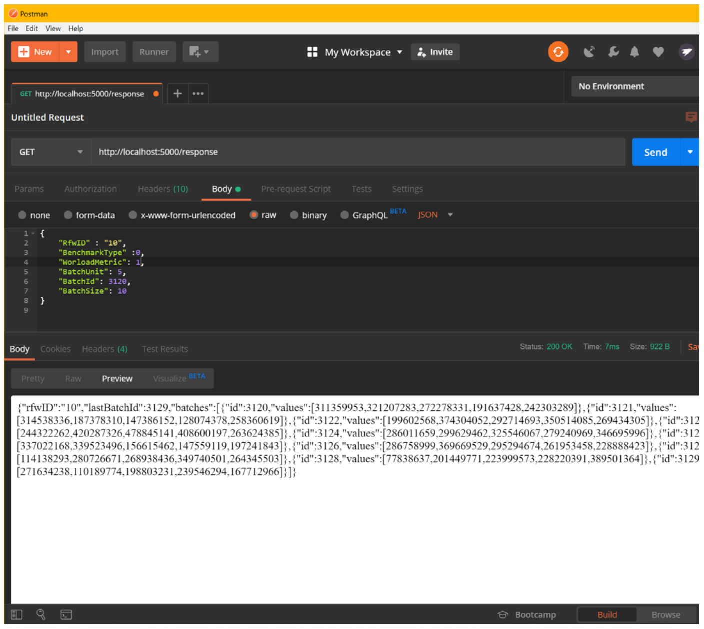
GET http://localhost:5000/response

Untitled Request

GET http://localhost:5000/response

```
{
    "RfwID" : "10",
    "BenchmarkType" :0,
    "WorloadMetric": 1,
    "BatchUnit": 5,
    "BatchId": 3120,
    "BatchSize": 10
}
```

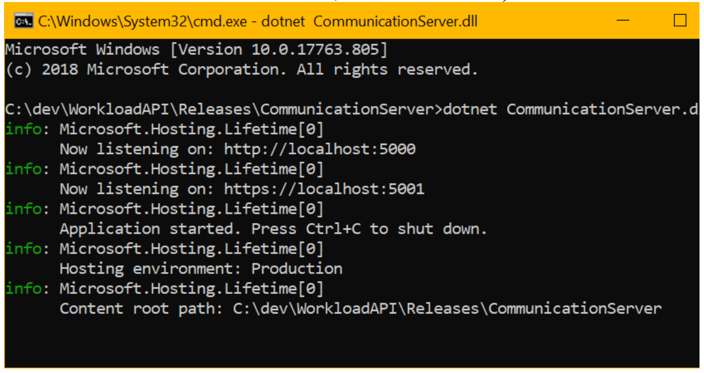
Figure 5: Making a request to the Rest API with Postman



```
{
    "RfwID" : "10",
    "BenchmarkType" :0,
    "WorloadMetric": 1,
    "BatchUnit": 5,
    "BatchId": 3120,
    "BatchSize": 10
}
```

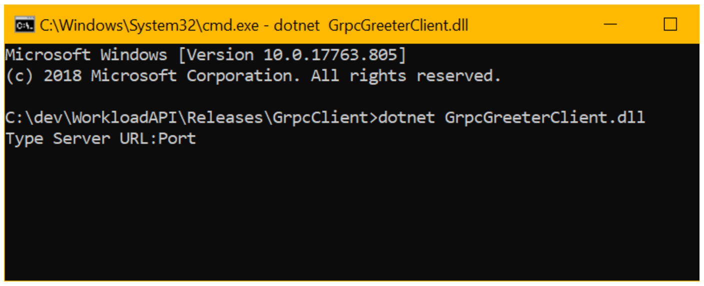
```
{"rfwID":"10","lastBatchId":3129,"batches":[{"id":3120,"values":[311359953,321207283,272278331,191637428,242303289]},{"id":3121,"values":
[314538336,187378310,147386152,128074378,258360619]},{"id":3122,"values":[199602568,374304052,292714693,350514085,269434305]},{"id":312
[244322262,420287326,478845141,408600197,263624385]},{"id":3124,"values":[286011659,299629462,325546067,279240969,346695996]},{"id":312
[337022168,339523496,156615462,147559119,197241843]},{"id":3126,"values":[286758999,369669529,295294674,261953458,228888423]},{"id":312
[114138293,280726671,268938436,349740501,264345503]},{"id":3128,"values":[77838637,201449771,223999573,228220391,389501364]},{"id":3129
[271634238,110189774,198803231,239546294,167712966]}]}
```

Figure 6 : Response from RESTAPI server when (RFW ID: 10, BenchmarkType: 0, WorkloadMetric: 1, Batch Unit: 5, Batch ID: 3120, Batch Size: 10)


C:\Windows\System32\cmd.exe - dotnet CommunicationServer.dll

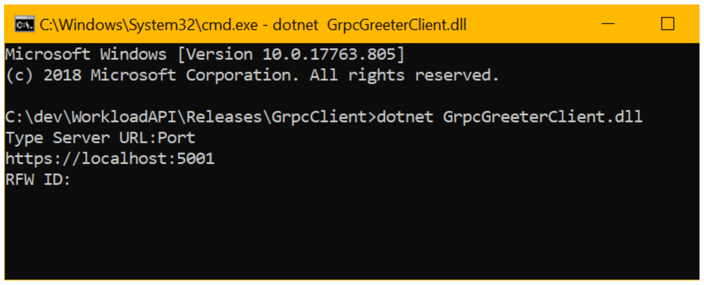
```
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\dev\WorkloadAPI\Releases\CommunicationServer>dotnet CommunicationServer.d
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: http://localhost:5000
info: Microsoft.Hosting.Lifetime[0]
      Now listening on: https://localhost:5001
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: C:\dev\WorkloadAPI\Releases\CommunicationServer
```

Figure 7: Running the gRPC server in console



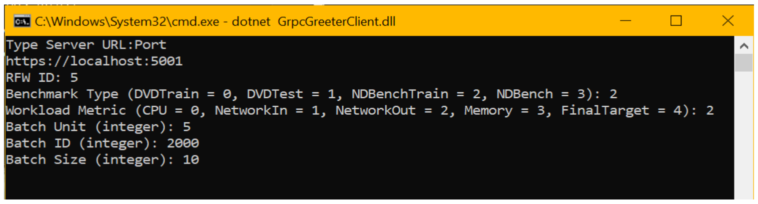
```
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\dev\WorkloadAPI\Releases\GrpcClient>dotnet GrpcGreeterClient.dll
Type Server URL:Port
```

Figure 8: Running the gRPC client in console



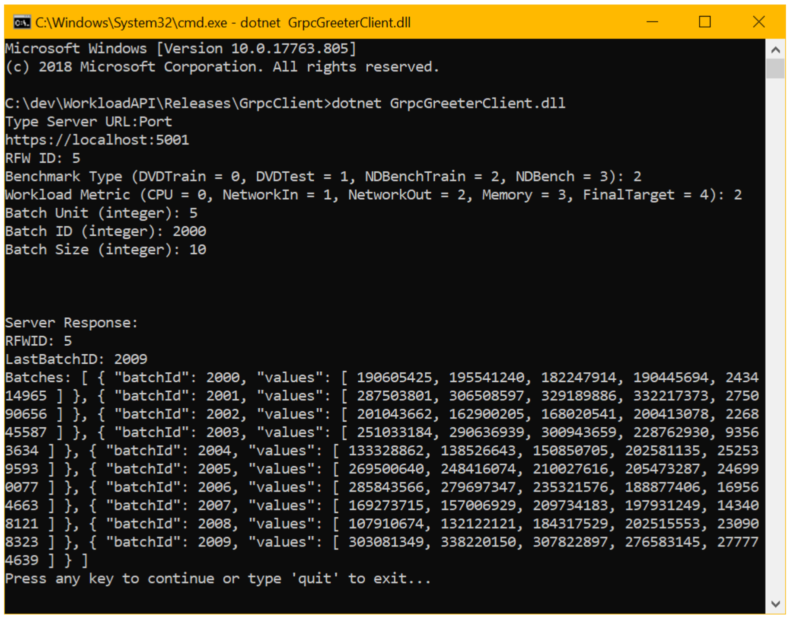
```
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\dev\WorkloadAPI\Releases\GrpcClient>dotnet GrpcGreeterClient.dll
Type Server URL:Port
https://localhost:5001
RFW ID:
```

Figure 9: Entering server URL with port number in gRPC client



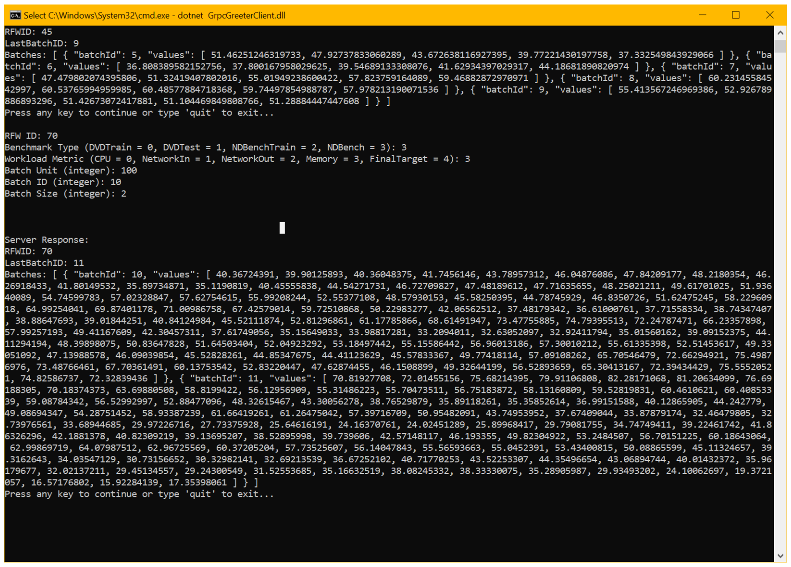
```
Type Server URL:Port
https://localhost:5001
RFW ID: 5
Benchmark Type (DVDTrain = 0, DVDTest = 1, NDBenchTrain = 2, NDBench = 3): 2
Workload Metric (CPU = 0, NetworkIn = 1, NetworkOut = 2, Memory = 3, FinalTarget = 4): 2
Batch Unit (integer): 5
Batch ID (integer): 2000
Batch Size (integer): 10
```

Figure 10: Entering Request information in gRPC client



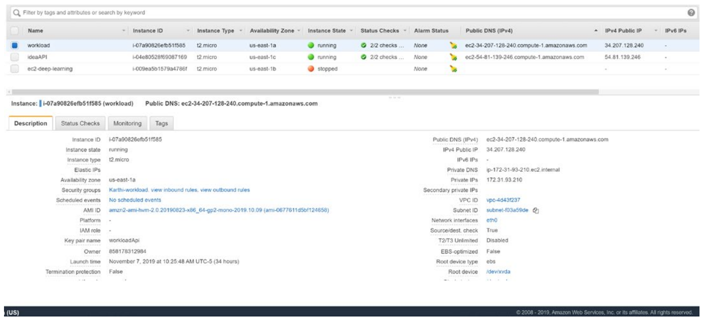
```
Microsoft Windows [Version 10.0.17763.805]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\dev\WorkloadAPI\Releases\GrpcClient>dotnet GrpcGreeterClient.dll
Type Server URL:Port
https://localhost:5001
RFW ID: 5
Benchmark Type (DVDTrain = 0, DVDTest = 1, NDBenchTrain = 2, NDBench = 3): 2
Workload Metric (CPU = 0, NetworkIn = 1, NetworkOut = 2, Memory = 3, FinalTarget = 4): 2
Batch Unit (integer): 5
Batch ID (integer): 2000
Batch Size (integer): 10



Server Response:
RFWID: 5
LastBatchID: 2009
Batches: [ { "batchId": 2000, "values": [ 190605425, 195541240, 182247914, 190445694, 2434
14965 ] }, { "batchId": 2001, "values": [ 287503801, 306508597, 329189886, 332217373, 2750
90656 ] }, { "batchId": 2002, "values": [ 201043662, 162900205, 168020541, 200413078, 2268
45587 ] }, { "batchId": 2003, "values": [ 251033184, 290636939, 300943659, 228762930, 9356
3634 ] }, { "batchId": 2004, "values": [ 133328862, 138526643, 150850705, 202581135, 25253
9593 ] }, { "batchId": 2005, "values": [ 269500640, 248416074, 210027616, 205473287, 24699
0077 ] }, { "batchId": 2006, "values": [ 285843566, 279697347, 235321576, 188877406, 16956
4663 ] }, { "batchId": 2007, "values": [ 169273715, 157006929, 209734183, 197931249, 14340
8121 ] }, { "batchId": 2008, "values": [ 107910674, 132122121, 184317529, 202515553, 23090
8323 ] }, { "batchId": 2009, "values": [ 303081349, 338220150, 307822897, 276583145, 27777
4639 ] } ]
Press any key to continue or type 'quit' to exit...
```

Figure 11 : Response from gRPC server when (RFW ID: 5, BenchmarkType: 2, WorkloadMetric: 2, Batch Unit: 5, Batch ID: 2000, Batch Size: 10)


Select C:\Windows\System32\cmd.exe - dotnet GrpcGreeterClient.dll

```
RFWID: 45
LastBatchID: 9
Batches: [ { "batchId": 5, "values": [ 51.46251246319733, 47.92737833060289, 43.672638116927395, 39.77221430197758, 37.332549843929066 ] }, { "ba
tchId": 6, "values": [ 36.808389582152756, 37.800167958029625, 39.54689133308076, 41.62934397029317, 44.18681890820974 ] }, { "batchId": 7, "valu
es": [ 47.479802074395806, 51.32419407802016, 55.01949238600422, 57.823759164089, 59.46882872970971 ] }, { "batchId": 8, "values": [ 60.231455845
42997, 60.5376599495985, 60.48577884718368, 59.74497854988787, 57.978213190071536 ] }, { "batchId": 9, "values": [ 55.413567246969386, 52.926789
886893296, 51.42673072417881, 51.104469849808766, 51.28884447447608 ] } ]
Press any key to continue or type 'quit' to exit...

RFW ID: 70
Benchmark Type (DVDTrain = 0, DVDTest = 1, NDBenchTrain = 2, NDBench = 3): 3
Workload Metric (CPU = 0, NetworkIn = 1, NetworkOut = 2, Memory = 3, FinalTarget = 4): 3
Batch Unit (integer): 100
Batch ID (integer): 10
Batch Size (integer): 2



Server Response:
RFWID: 70
LastBatchID: 11
Batches: [ { "batchId": 10, "values": [ 40.36724391, 39.90125893, 40.36048375, 41.7456146, 43.78957312, 46.04876086, 47.84209177, 48.2180354, 46.
26918433, 41.80149532, 35.89734871, 35.1190819, 40.45555838, 44.54271731, 46.72709827, 47.48189612, 47.71635655, 48.25021211, 49.61701025, 51.936
40089, 54.74599783, 57.02328847, 57.62754615, 55.99208244, 52.55377108, 48.57930153, 45.58250395, 44.78745929, 46.8350726, 51.62475245, 58.229609
18, 64.99254041, 69.87401178, 71.00986758, 67.42579014, 59.72510868, 50.22983277, 42.06562512, 37.48179342, 36.61000761, 37.71558334, 38.74347407
, 38.88647693, 39.01844251, 40.84124984, 45.52111874, 52.81296861, 61.17785866, 68.61491947, 73.47755885, 74.79395513, 72.24787471, 66.23357898,
57.99257193, 49.41167609, 42.30457311, 37.61749056, 35.15649033, 33.98817281, 33.2094011, 32.63052097, 32.92411794, 35.01560162, 39.09152375, 44.
11294194, 48.39898075, 50.83647828, 51.64503404, 52.04923292, 53.18497442, 55.15586442, 56.96013186, 57.30010212, 55.61335398, 52.51453617, 49.33
051092, 47.13988578, 46.09039854, 45.52828261, 44.85347675, 44.41123629, 45.57833367, 49.77418114, 57.09108262, 65.70546479, 72.66294921, 75.4987
6976, 73.48766461, 67.70361491, 60.13753542, 52.83220447, 47.62874455, 46.1508899, 49.32644199, 56.52893659, 65.30413167, 72.39434429, 75.5552052
1, 74.82586737, 72.32839436 ] }, { "batchId": 11, "values": [ 70.81927708, 72.01455156, 75.68214395, 79.91106808, 82.28171068, 81.20634099, 76.69
188305, 70.18374373, 63.69880508, 58.8199422, 56.12956909, 55.31486223, 55.70473511, 56.75183872, 58.13160809, 59.52819831, 60.4610621, 60.408533
39, 59.08784342, 56.52992997, 52.88477096, 48.32615467, 43.30056278, 38.76529879, 35.89118261, 35.35852614, 36.99151588, 40.12865905, 44.242779,
49.08694347, 54.28751452, 58.93387239, 61.66419261, 61.26475042, 57.39716709, 50.95482091, 43.74953952, 37.67409044, 33.87879174, 32.46479805, 32
.73976561, 33.68944685, 29.97226716, 27.73375928, 25.64616191, 24.16370761, 24.02451289, 25.89968417, 29.79081755, 34.74749411, 39.22461742, 41.8
6326296, 42.1881378, 40.82309219, 39.13695207, 38.52895998, 39.739606, 42.57148117, 46.193355, 49.82304922, 53.2484507, 56.70151225, 60.18643064,
 62.99869719, 64.07987512, 62.96725569, 60.37205204, 57.73525607, 56.14047843, 55.56593663, 55.0452391, 53.43400815, 50.08865599, 45.11324657, 39
.3162643, 34.03547129, 30.73156652, 30.32982141, 32.69213539, 36.67252102, 40.71770253, 43.52253307, 44.35496654, 43.06894744, 40.01432372, 35.96
179677, 32.02137211, 29.45134557, 29.24300549, 31.52553685, 35.16632519, 38.08245332, 38.33330075, 35.28905987, 29.93493202, 24.10062697, 19.3721
057, 16.57176802, 15.92284139, 17.35398061 ] } ]
Press any key to continue or type 'quit' to exit...
```

Figure 12 : Response from gRPC server when (RFW ID:70, BenchmarkType: 3, WorkloadMetric: 3, Batch Unit: 100, Batch ID: 10, Batch Size: 2)



Filter by tags and attributes or search by keyword

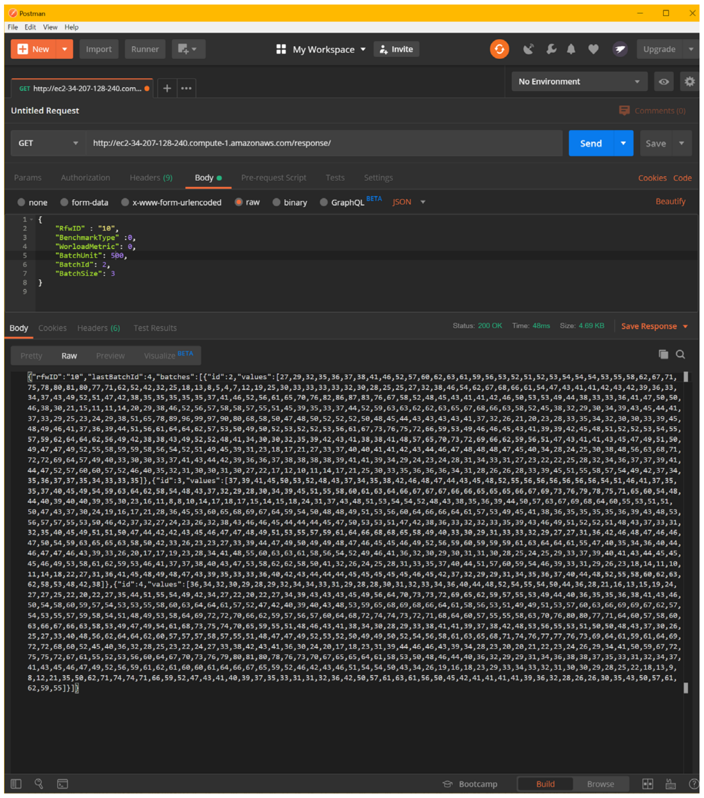
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
workload	i-07a90826efb51f585	t2.micro	us-east-1a	running	2/2 checks ...	None	ec2-34-207-128-240.compute-1.amazonaws.com	34.207.128.240	-
ideaAPI	i-04e80528f69087169	t2.micro	us-east-1c	running	2/2 checks ...	None	ec2-54-81-139-246.compute-1.amazonaws.com	54.81.139.246	-
ec2-deep-learning	i-009ea5b1579a4786f	t2.micro	us-east-1b	stopped		None		-	-

Instance: i-07a90826efb51f585 (workload) Public DNS: ec2-34-207-128-240.compute-1.amazonaws.com

Instance ID	i-07a90826efb51f585	Public DNS (IPv4)	ec2-34-207-128-240.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	34.207.128.240
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs		Private DNS	ip-172-31-93-210.ec2.internal
Availability zone	us-east-1a	Private IPs	172.31.93.210
Security groups	Karthi-workload. view inbound rules. view outbound rules	Secondary private IPs	
Scheduled events	No scheduled events	VPC ID	vpc-4d43f237
AMI ID	amzn2-ami-hvm-2.0.20190823-x86_64-gp2-mono-2019.10.09 (ami-0677611d5bf124658)	Subnet ID	subnet-f03a59de
Platform	-	Network interfaces	eth0
IAM role	-	Source/dest. check	True
Key pair name	workloadApi	T2/T3 Unlimited	Disabled
Owner	858178312984	EBS-optimized	False
Launch time	November 7, 2019 at 10:25:48 AM UTC-5 (34 hours)	Root device type	ebs
Termination protection	False	Root device	/dev/xvda

(US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Figure 13: AWS Console showing ec2 instance running with public DNS



GET http://ec2-34-207-128-240.compute-1.amazonaws.com/response/

```
{
    "RfwID" : "10",
    "BenchmarkType" :0,
    "WorloadMetric": 0,
    "BatchUnit": 500,
    "BatchId": 2,
    "BatchSize": 3
}
```

Status: 200 OK Time: 48ms Size: 4.69 KB

```
{"rfwID":"10","lastBatchId":4,"batches":[{"id":2,"values":[27,29,32,35,36,37,38,41,46,52,57,60,62,63,61,59,56,53,52,51,52,53,54,54,54,53,55,58,62,67,71,
75,78,80,81,80,77,71,62,52,42,32,25,18,13,8,5,4,7,12,19,25,30,33,33,33,33,32,30,28,25,25,27,32,38,46,54,62,67,68,66,61,54,47,43,41,41,42,43,42,39,36,33,
34,37,43,49,52,51,47,42,38,35,35,35,35,35,37,41,46,52,56,61,65,70,76,82,86,87,83,76,67,58,52,48,45,43,41,41,42,46,50,53,53,49,44,38,33,33,36,41,47,50,50,
46,38,30,21,15,11,11,14,20,29,38,46,52,56,57,58,58,57,55,51,45,39,35,33,37,44,52,59,63,63,62,62,63,65,67,68,66,63,58,52,45,38,32,29,30,34,39,43,45,44,41,
37,33,29,25,23,24,29,38,51,65,78,89,96,99,97,90,80,68,58,50,47,48,50,52,52,52,50,48,45,44,43,43,43,43,41,37,32,26,21,20,23,28,33,35,34,32,30,30,33,39,45,
48,49,46,41,37,36,39,44,51,56,61,64,64,62,57,53,50,49,50,52,53,52,52,53,56,61,67,73,76,75,72,66,59,53,49,46,46,45,43,41,39,39,42,45,48,51,52,52,53,54,55,
57,59,62,64,64,62,56,49,42,38,38,43,49,52,52,48,41,34,30,30,32,35,39,42,43,41,38,38,41,48,57,65,70,73,72,69,66,62,59,56,51,47,43,41,41,43,45,47,49,51,50,
49,47,47,49,52,55,58,59,59,58,56,54,52,51,49,45,39,31,23,18,17,21,27,33,37,40,40,41,41,42,43,44,46,47,48,48,48,47,45,40,34,28,24,25,30,38,48,56,63,68,71,
72,72,69,64,57,49,40,33,30,30,33,37,41,43,44,42,39,36,36,37,38,38,38,38,39,41,41,39,34,29,24,23,24,28,31,34,33,31,27,23,22,22,25,28,32,34,36,37,37,39,41,
44,47,52,57,60,60,57,52,46,40,35,32,31,30,30,31,30,27,22,17,12,10,11,14,17,21,25,30,33,35,36,36,36,34,31,28,26,26,28,33,39,45,51,55,58,57,54,49,42,37,34,
35,36,37,37,35,34,33,33,35]},{"id":3,"values":[37,39,41,45,50,53,52,48,43,37,34,35,38,42,46,48,47,44,43,45,48,52,55,56,56,56,56,56,56,54,51,46,41,37,35,
35,37,40,45,49,54,59,63,64,62,58,54,48,43,37,32,29,28,30,34,39,45,51,55,58,60,61,63,64,66,67,67,67,66,66,65,65,65,66,67,69,73,76,79,78,75,71,65,60,54,48,
44,40,39,40,40,39,35,30,23,16,11,8,8,10,14,17,18,17,15,14,15,18,24,31,37,43,48,51,53,54,54,52,48,43,38,35,36,39,44,50,57,63,67,69,68,64,60,55,53,51,51,
50,47,43,37,30,24,19,16,17,21,28,36,45,53,60,65,68,69,67,64,59,54,50,48,48,49,51,53,56,60,64,66,66,64,61,57,53,49,45,41,38,36,35,35,35,35,36,39,43,48,53,
56,57,57,55,53,50,46,42,37,32,27,24,23,26,32,38,43,46,46,45,44,44,44,45,47,50,53,53,51,47,42,38,36,33,32,32,33,35,39,43,46,49,51,52,52,51,48,43,37,33,31,
32,35,40,45,49,51,51,50,47,44,42,42,43,45,46,47,47,48,49,51,53,55,57,59,61,64,66,68,68,65,58,49,40,33,30,29,31,33,33,32,29,27,27,31,36,42,46,48,47,46,46,
47,50,54,59,63,65,65,63,58,50,42,33,26,23,23,27,33,39,44,47,49,50,49,49,48,47,46,45,45,46,49,52,56,59,60,59,59,59,61,63,64,64,61,55,47,40,35,34,36,40,44,
46,47,47,46,43,39,33,26,20,17,17,19,23,28,34,41,48,55,60,63,63,61,58,56,54,52,49,46,41,36,32,30,29,30,31,31,30,28,25,24,25,29,33,37,39,40,41,43,44,45,45,
45,46,49,53,58,61,62,59,53,46,41,37,37,38,40,43,47,53,58,62,62,58,50,41,32,26,24,25,28,31,33,35,37,40,44,51,57,60,59,54,46,39,33,31,29,26,23,18,14,11,10,
11,14,18,22,27,31,36,41,45,48,49,48,47,43,39,35,33,33,36,40,42,43,44,44,44,45,45,45,45,45,46,45,42,37,32,29,29,31,34,35,36,37,40,44,48,52,55,58,60,62,63,
62,58,53,48,42,38]},{"id":4,"values":[36,34,32,30,29,28,29,32,34,34,33,31,29,28,28,30,31,32,33,34,36,40,44,48,52,54,55,54,50,44,36,28,21,16,13,15,19,24,
27,27,25,22,20,22,27,35,44,51,55,54,49,42,34,27,22,20,22,27,34,39,43,43,43,45,49,56,64,70,73,73,72,69,65,62,59,57,55,53,49,44,40,36,35,35,36,38,41,43,46,
50,54,58,60,59,57,54,53,53,55,58,60,63,64,64,61,57,52,47,42,40,39,40,43,48,53,59,65,68,69,68,66,64,61,58,56,53,51,49,49,51,53,57,60,63,66,69,69,67,62,57,
54,53,55,57,59,58,54,51,48,49,53,58,64,69,72,72,70,66,62,59,57,56,57,60,64,68,72,74,74,73,72,71,68,64,60,57,55,55,58,63,70,76,80,80,77,71,64,60,57,58,60,
63,66,67,66,63,58,53,49,47,49,54,61,68,73,75,74,70,65,59,55,51,48,46,43,41,38,34,30,28,29,33,38,41,41,39,37,38,42,48,53,56,55,53,51,50,50,48,43,37,30,26,
25,27,33,40,48,56,62,64,64,62,60,57,57,57,58,57,55,51,48,47,47,49,52,53,52,50,49,49,50,52,54,56,58,61,63,65,68,71,74,76,77,77,76,73,69,64,61,59,61,64,69,
72,72,68,60,52,45,40,36,32,28,25,23,22,24,27,33,38,42,43,41,36,30,24,20,17,18,23,31,39,44,46,46,43,39,34,28,23,20,20,21,22,23,24,26,29,34,41,50,59,67,72,
75,75,72,67,61,55,52,53,56,60,64,67,70,73,76,79,80,81,80,78,76,73,70,67,65,65,64,61,58,53,50,48,46,44,40,36,32,29,29,31,34,36,38,38,37,35,33,31,32,34,37,
41,43,45,46,47,49,52,56,59,61,62,61,60,60,61,64,66,67,65,59,52,46,42,43,46,51,54,54,50,43,34,26,19,16,18,23,29,33,34,33,32,31,30,30,29,28,25,22,18,13,9,
8,12,21,35,50,62,71,74,74,71,66,59,52,47,43,41,40,39,37,35,33,31,31,32,36,42,50,57,61,63,61,56,50,45,42,41,41,41,41,39,36,32,28,26,26,30,35,43,50,57,61,
62,59,55]}]}
```

Figure 14: Response of REST API hosted on ec2 instance

IX. References

[1] "gRPC Concepts," [Online]. Available: https://grpc.io/docs/guides/concepts/. [Accessed 02 11 2019].

[2] R. Anderson and M. Wasson, "Tutorial: Create a web API with ASP.NET Core," Microsoft, 28 09 2019. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.0&tabs=visual-studio. [Accessed 01 11 2019].

[3] "Try the new System.Text.Json APIs," Microsoft, 13 06 2019. [Online]. Available: https://devblogs.microsoft.com/dotnet/try-the-new-system-text-json-apis/. [Accessed 01 11 2019].

[4] "REST and gRPC," Microsoft, 09 07 2019. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/rest-grpc. [Accessed 01 11 2019].

[5] T. Rothschild , "Protocol Buffers, Part 1 — Serialization Library for Microservices," Codeburst, 06 07 2018. [Online]. Available: https://codeburst.io/protocol-buffers-part-1-serialization-library-for-microservices-37418e72908b. [Accessed 01 11 2019].

[6] "Developer Guide," Google , [Online]. Available: https://developers.google.com/protocol-buffers/docs/overview. [Accessed 01 11 2019].

[7] "gRPC for .NET," github, [Online]. Available: https://github.com/grpc/grpc-dotnet. [Accessed 28 10 2019].

[8] J. Luo and J. N.-. King, "Introduction to gRPC on .NET Core," Microsoft, 19 09 2019. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/grpc/?view=aspnetcore-3.0. [Accessed 27 10 2019].

[9] J. N.-. King, "Troubleshoot gRPC on .NET Core," Microsoft, 15 10 2019. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/grpc/troubleshoot?view=aspnetcore-3.0&branch=master#unable-to-start-aspnet-core-grpc-app-on-macos. [Accessed 28 10 2019].