openGemini开发实践实验

实验描述和目的

本实验介绍在openGemini时序数据库中开发一个hello新函数，围绕新功能如何添加单元测试和功能测试用例，以及分析函数性能的方法。目的是熟悉数据库开发、调试、功能测试、性能分析等全部流程

开发环境搭建

go version v1.18 及以上

Windows编译需要单独安装gcc，参考链接如下

https://www.cnblogs.com/kala00k/p/16364116.html

python version v3.7 及以上

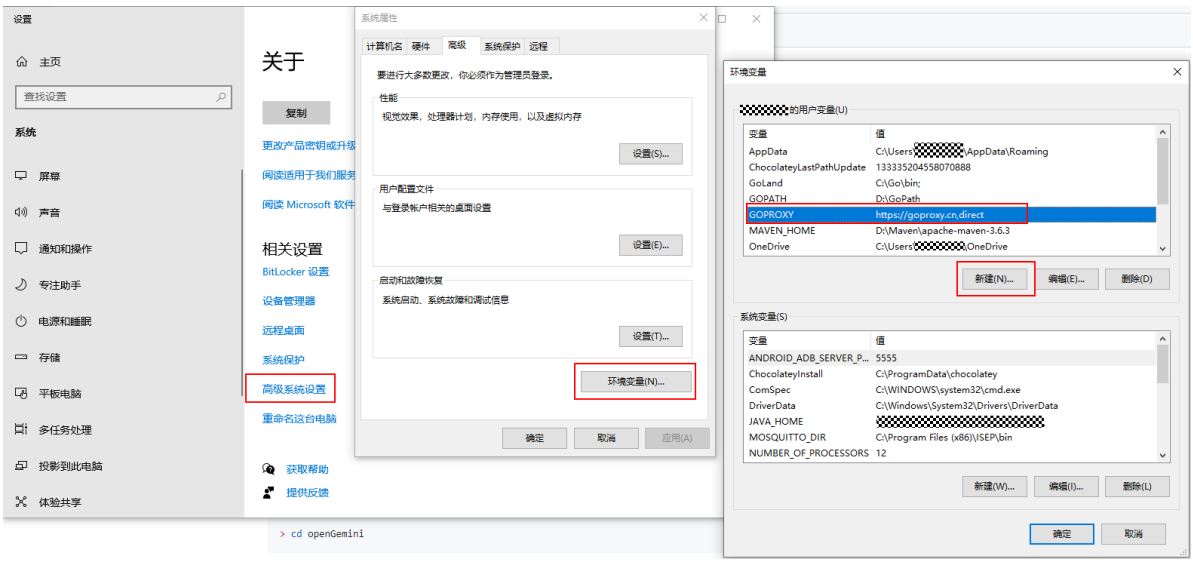
Go环境变量设置

Linux

```
export GOPATH=/path/to/dir
export GOBIN=$GOPATH/bin
export GO111MODULE=on
export GONOSUMDB=*
export GOSUMDB=off
export GOPROXY=https://goproxy.cn,direct
```

windows

”我的电脑“ --> "属性" --> ”高级系统设置“ --> ”环境变量" ,添加如下环境变量



```
GOPROXY = https://goproxy.cn,direct
GOPATH=D:\GoPath
GO111MODULE=on

//亦可通过命令
> go env -w GOPROXY=https://goproxy.cn,direct
> go env -w GOPATH=D:\GoPath
> go env -w GO111MODULE=on
```

查看环境变量是否设置成功

```
> go env
C:\Users\xxx>go env
set GO111MODULE=on
set GOARCH=amd64
set GOBIN=
set GOCACHE=C:\Users\xxx\AppData\Local\go-build
set GOENV=C:\Users\xxx\AppData\Roaming\go\env
set GOEXE=.exe
set GOEXPERIMENT=
set GOFLAGS=
set GOHOSTARCH=amd64
set GOHOSTOS=windows
set GOINSECURE=
set GOMODCACHE=D:\GoPath\pkg\mod
set GONOPROXY=
set GONOSUMDB=*
set GOOS=windows
set GOPATH=D:\GoPath
set GOPRIVATE=
set GOPROXY=https://goproxy.cn,direct
set GOROOT=C:\Program Files\Go
set GOSUMDB=off
set GOTMPDIR=
set GOTOOLDIR=C:\Program Files\Go\pkg\tool\windows_amd64
set GOVCS=
set GOVERSION=go1.18.10
set GCCGO=gccgo
set GOAMD64=v1
set AR=ar
set CC=gcc
set CXX=g++
set CGO_ENABLED=1
set GOMOD=NUL
set GOWORK=
set CGO_CFLAGS=-g -O2
set CGO_CPPFLAGS=
set CGO_CXXFLAGS=-g -O2
set CGO_FFLAGS=-g -O2
set CGO_LDFLAGS=-g -O2
set PKG_CONFIG=pkg-config
set GOGCCFLAGS=-m64 -mthreads -fmessage-length=0 -fdebug-prefix-
map=C:\Users\xxx\AppData\Local\Temp\go-build1782990979=/tmp/go-build -gno-record-
gcc-switches
```

编译

代码下载

```
//from github
> git clone https://github.com/openGemini/openGemini.git
//from gitee mirror
> git clone https://gitee.com/mirrors/opengemini.git
```

编译

```
//所有平台编译方法一致
> python build.py
```

运行(windows)

```
> cd build
> ./ts-server.exe
2023/07/26 11:29:56 proto: duplicate proto type registered:
influxql.Measurements
2023/07/26 11:29:56 proto: duplicate proto type registered: influxql.Measurement

 _________   ______    ______   ________  _______  ____   ____  ________
_______
|  _   _  |.' ____ \ .' ____ \ |_   __  ||_   __ \|_  _| |_  _||_   __  ||_   __
\
|_/ | | \_|| (___ \_|| (___ \_|  | |_ \_|  | |__) | \ \   / /    | |_ \_|  | |__)
|
    | |     _.____''.  _.____''.   |  _| _   |  _/   \ \ / /     |  _| _   |  __
/
   _| |_   | \____) || \____) | _| |__/ | _| |  \ \_  \ ' /     _| |__/ | _| |  \
\_
  |_____|   \_______.' \_______.'|________||____| |___|  \_/     |________||____|
|___|

2023-07-26 11:29:56.1832582 +0800 CST m=+0.465951301 TSMeta starting
2023-07-26 11:29:58.1054196 +0800 CST m=+2.388116701 TSStore starting
successfully opened storage "C:\\Users\\xxx\\.openGemini\\data" in 0.005 seconds
2023-07-26 11:29:58.1123728 +0800 CST m=+2.395069901 TSSQL starting
```

连接

```
> ./ts-cli.exe
2023/07/26 11:30:25 proto: duplicate proto type registered:
influxql.Measurements
2023/07/26 11:30:25 proto: duplicate proto type registered: influxql.Measurement
openGemini CLI 0.1.0 (rev-revision)
Please use `quit`, `exit` or `Ctrl-D` to exit this program.
> show databases
name: databases
+------+
| name |
+------+
1 columns, 0 rows in set
> create database db
> use db
> insert mst,host="192.168.1.1" cpu=0.12,mem=0.32
```

```
> select * from mst
name: mst
+--------------------+------+--------------+------+
| time               | cpu  | host         | mem  |
+--------------------+------+--------------+------+
| 1690342471346562300 | 0.12 | "192.168.1.1" | 0.32 |
+--------------------+------+--------------+------+
4 columns, 1 rows in set
```

到此，所有开发环境和准备工作已经就绪（注：在linux环境上，运行方式相同）

开发hello算子

在这里，我们将实现一个HELLO()函数，该函数有一个字符串参数。

```
> select hello("name") from mst
ERR: undefined function hello()
```

第一步

在open_src/influx/query/compile.go, 修改isStringFunction方法，添加**hello**函数名称

```
func isStringFunction(call *influxql.Call) bool {
    switch call.Name {
    case "str", "strlen", "substr", "hello":
        return true
    }
    return false
}
```

在engine/executor/schema.go, 修改isStringFunction方法，添加**hello**函数名称

```
func (qs *QuerySchema) isStringFunction(call *influxql.Call) bool {
    switch call.Name {
    case "str", "strlen", "substr", "hello":
        return true
}
```

第二步

设置hello函数的调用类型，修改open_src/influx/query/functions.go CallType方法

```
func (m StringFunctionTypeMapper) CallType(name string, _ []influxql.DataType)
(influxql.DataType, error) {
    switch name {
    case "str":
        return influxql.Boolean, nil
    case "strlen":
        return influxql.Integer, nil
    case "substr":
        return influxql.String, nil
    case "hello":
        return influxql.String, nil
    default:
        return influxql.Unknown, nil
```

```
    }
}
```

第三步

实现hello方法主体功能

```
func (v StringValuer) Call(name string, args []interface{}) (interface{}, bool)
{
    switch name {
    case "strlen":
        ...
    case "str":
        ...
    case "substr":
        ...
    case "hello":
        if len(args) != 1 {
            return nil, false
        }
        if arg0, ok := args[0].(string); ok {
            return HelloFunc(arg0), true
        }
        return nil, true
    default:
        return nil, false
    }
}
func HelloFunc(srcStr string) string {
    // 测试性能优化时放开下面注释
    // var h []byte
    // h = make([]byte, 200*1024*1024)
    // fmt.Fprintf(os.Stdout,"%v",h)
    return "hello, " + srcStr
}
```

第四步

重新编译，测试新功能

```
> insert mst,author="xiangyu" msg="world"
> SELECT hello(msg) from mst
+----------------------+------------------+
| time                 | hello            |
+----------------------+------------------+
| 2021-08-16T16:00:00Z | hello, world     |
+----------------------+------------------+

> insert mst,author="zhangsan" msg="database"
> SELECT author,hello(msg) as say from mst
name: mst
+----------------------+------------+-----------------+
| time                 | author     | say             |
+----------------------+------------+-----------------+
| 1690422898225398100  | "xiangyu"  | hello, world    |
| 1690423211728874800  | "zhangsan" | hello, database |
+----------------------+------------+-----------------+
```

```
3 columns, 3 rows in set
```

数据写入参考文档

https://docs.opengemini.org/zh/guide/geminiql/sql_syntax/DML/insert.html

单元测试

我们需要测试engine/executor/string_functions.go中的**HelloFunc**是否符合预期：以**hello开头。**

在engine/executor/string_function_test.go文件中，添加如下测试：

```
func TestStringFunctionHello(t *testing.T) {
    stringValuer := executor.StringValuer{}
    inputName := "hello"
    inputArgs := []interface{}{"Alice", "Bob", "Carry"}
    expects := []interface{}{"hello, Alice", "hello, Bob", "hello, Carry"}
    outputs := make([]interface{}, 0, len(expects))
    for _, arg := range inputArgs {
        if out, ok := stringValuer.Call(inputName, []interface{}{arg}); ok {
            outputs = append(outputs, out)
        }
    }
    assert.Equal(t, outputs, expects)
}
```

集成测试

在tests/server_test.go文件中，增加如下测试函数：

```
func TestServer_Query_Aggregate_For_Hello_Functions(t *testing.T) {
    t.Parallel()
    s := OpenServer(NewParseConfig(testCfgPath))
    defer s.Close()

    if err := s.CreateDatabaseAndRetentionPolicy("db0",
NewRetentionPolicySpec("rp0", 1, 0), true); err != nil {
        t.Fatal(err)
    }

    writes := []string{
        fmt.Sprintf(`mst,country=china,name=azhu
age=12.3,height=70i,address="shenzhen",alive=TRUE 1629129600000000000`),
        fmt.Sprintf(`mst,country=american,name=alan
age=20.5,height=80i,address="shanghai",alive=FALSE 1629129601000000000`),
        fmt.Sprintf(`mst,country=germany,name=alang
age=3.4,height=90i,address="beijin",alive=TRUE 1629129602000000000`),
        fmt.Sprintf(`mst,country=japan,name=ahui
age=30,height=121i,address="guangzhou",alive=FALSE 1629129603000000000`),
        fmt.Sprintf(`mst,country=canada,name=aqiu
age=35,height=138i,address="chengdu",alive=TRUE 1629129604000000000`),
        fmt.Sprintf(`mst,country=china,name=agang
age=48.8,height=149i,address="wuhan" 1629129605000000000`),
        fmt.Sprintf(`mst,country=american,name=agan
age=52.7,height=153i,alive=TRUE 1629129606000000000`),
        fmt.Sprintf(`mst,country=germany,name=alin
age=28.3,address="anhui",alive=FALSE 1629129607000000000`),
```

```
        fmt.Sprintf(`mst,country=japan,name=ali
height=179i,address="xian",alive=TRUE 1629129608000000000`),
        fmt.Sprintf(`mst,country=canada
age=60.8,height=180i,address="hangzhou",alive=FALSE 1629129609000000000`),
        fmt.Sprintf(`mst,name=ahuang
age=102,height=191i,address="nanjin",alive=TRUE 1629129610000000000`),
        fmt.Sprintf(`mst,country=china,name=ayin
age=123,height=203i,address="zhengzhou",alive=FALSE 1629129611000000000`),
    }
    test := NewTest("db0", "rp0")
    test.writes = Writes{
        &Write{data: strings.Join(writes, "\n")},
    }

    test.addQueries([]*Query{
        &Query{
            name:    "SELECT hello(address)",
            command: `SELECT hello("address") FROM db0.rp0.mst`,
            exp:     `{"results":[{"statement_id":0,"series":
[{"name":"mst","columns":["time","hello"],"values":[["2021-08-
16T16:00:00Z","hello, shenzhen"],["2021-08-16T16:00:01Z","hello, shanghai"],
["2021-08-16T16:00:02Z","hello, beijin"],["2021-08-16T16:00:03Z","hello,
guangzhou"],["2021-08-16T16:00:04Z","hello, chengdu"],["2021-08-
16T16:00:05Z","hello, wuhan"],["2021-08-16T16:00:07Z","hello, anhui"],["2021-08-
16T16:00:08Z","hello, xian"],["2021-08-16T16:00:09Z","hello, hangzhou"],["2021-
08-16T16:00:10Z","hello, nanjin"],["2021-08-16T16:00:11Z","hello,
zhengzhou"]]}]}}`,
        },
    }...)

    for i, query := range test.queries {
        t.Run(query.name, func(t *testing.T) {
            if i == 0 {
                if err := test.init(s); err != nil {
                    t.Fatalf("test init failed: %s", err)
                }
            }
            if query.skip {
                t.Skipf("SKIP:: %s", query.name)
            }
            if err := query.Execute(s); err != nil {
                t.Error(query.Error(err))
            } else if !query.success() {
                t.Error(query.failureMessage())
            }
        })
    }
}
```

性能分析(profiling)

对于任何数据库系统来说，性能始终很重要。如果你想知道性能瓶颈在哪里，可以使用一个强大的 Go 分析工具，名为pprof

> 通过命令安装：go install github.com/google/pprof@latest
>
> 安装成功后，pprof程序在$GOPATH/bin目录下

> 安装graphviz，用于web显示火焰图。https://www.graphviz.org/download/

```
//打开pprof配置
[http]
pprof-enabled = true
```

第一步：打开**HelloFunc**中注释的用于构造性能缺陷的代码

第二步：重新编译，并启动ts-server.exe

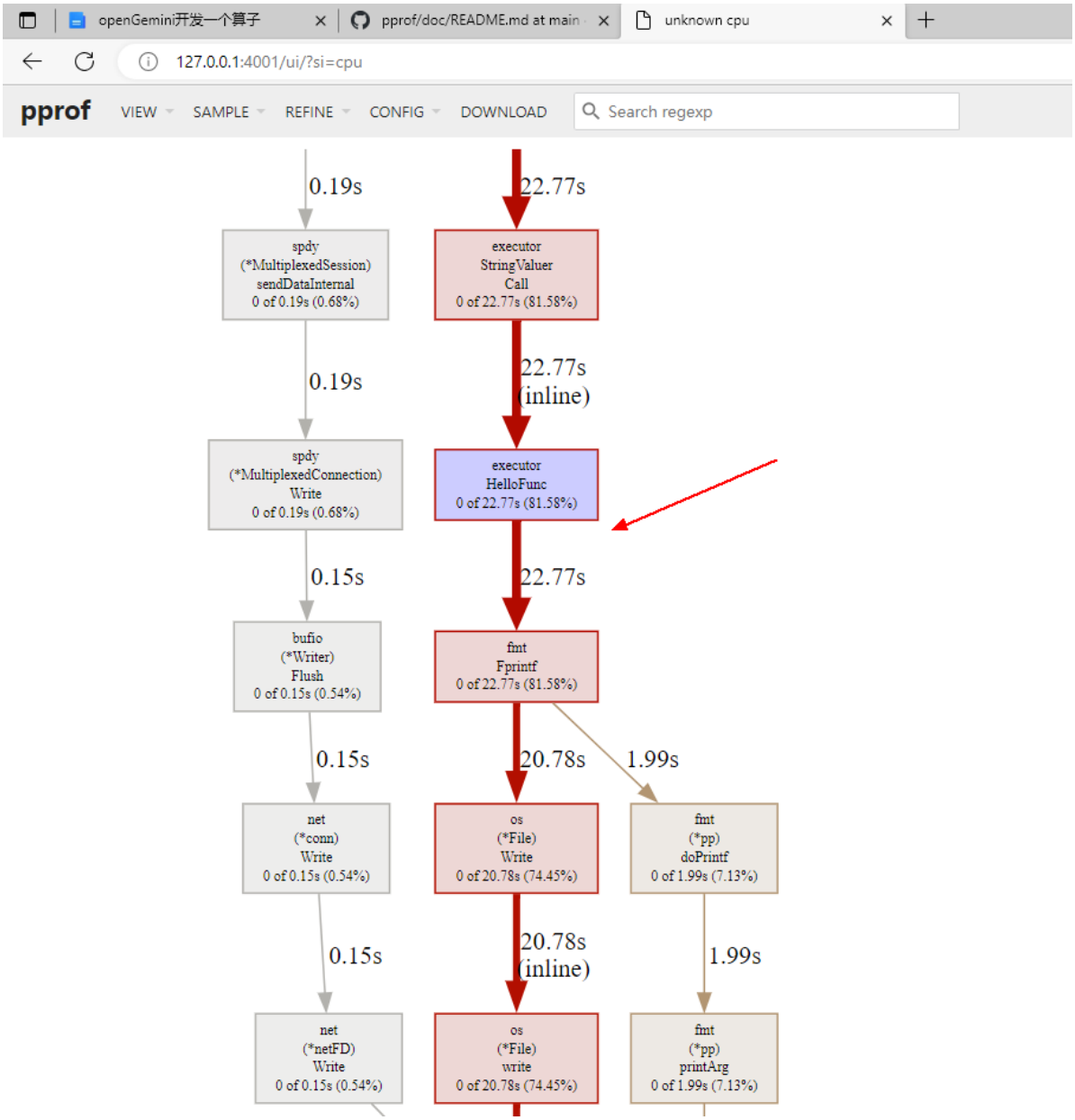
第三步：性能采样

```
curl -G "http://127.0.0.1:6061/debug/pprof/profile?seconds=45" > profile.profile
```

第四步：执行查询语句 “SELECT hello(msg) from mst”，等待采样工具结束

第五步：结果分析，可以看到HelloFunc方法执行时间太长了

```
go tool pprof -http 127.0.0.1:4001 profile.profile
```



参考链接

Flame Graphs (brendangregg.com)