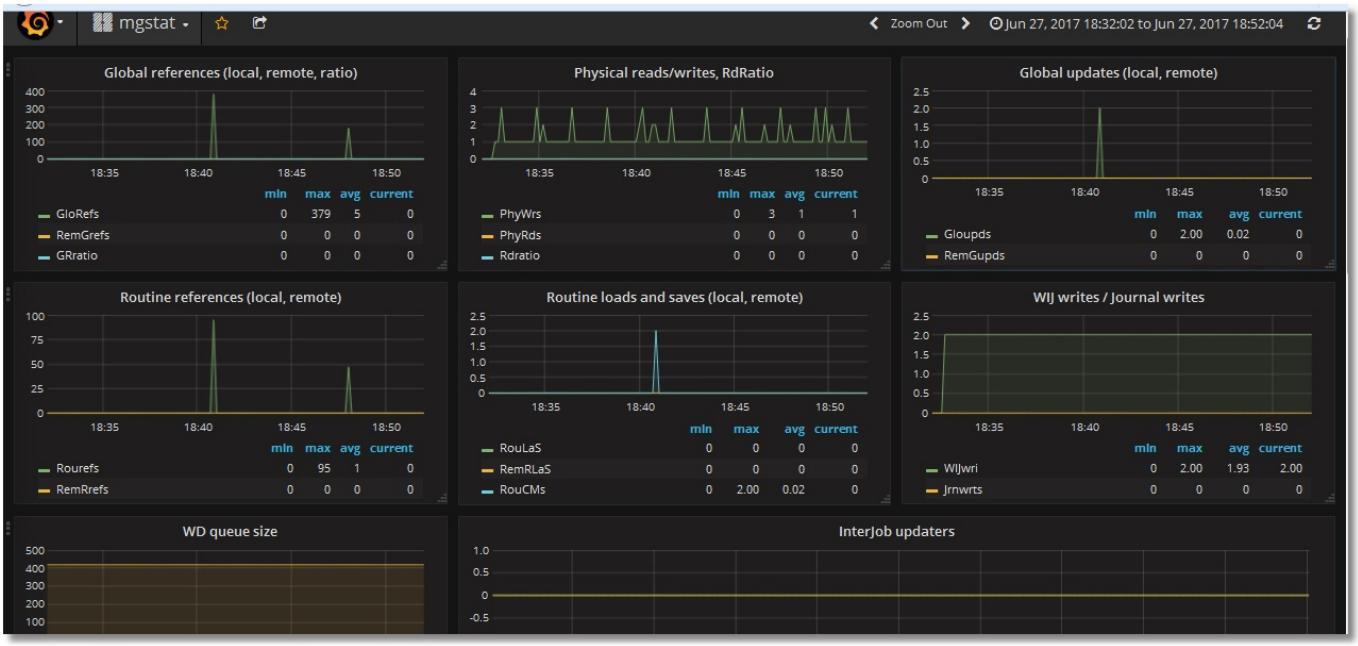
記事
Toshihiko Minamoto · 2020年11月11日 22m read

Grafana ベースの mgstat（InterSystems Caché / Ensemble / HealthShareのシステム監視ツール）用 GUI

こんにちは！ この記事は「Prometheus で InterSystems Caché を監視する」の続きになります。 ここでは ^mgstat ツールの動作結果を視覚化する方法を見ていきます。 このツールを使用すると、Caché のパフォーマンス統計、具体的なグローバルとルーチンの呼び出し数（ローカルおよびECP 経由）、書き込みデーモンのキュー長、ディスクに保存されるブロックと読み取られるブロックの数、ECP トラフィックの量などを取得できます。 ^mgstat は（対話的に、またはジョブによって）単独で起動したり、別のパフォーマンス測定ツールである ^pButtons と並行して起動したりできます。

ここでは 2 つのパートに分けて説明したいと思います。最初のパートでは ^mgstat によって収集された統計を図示し、2 番目のパートではこの統計を正確に収集する方法を集中して取り上げます。手短に言えば、ここでは $zu関数 を使用しています。 ただし、SYS.Stats パッケージのクラス経由で収集できる大部分のパラメーターに対応したオブジェクトインターフェースがあります。 ^mgstat に表示されるのは、収集できるパラメーターのほんの一部です。 その後、Grafana ダッシュボードですべてのパラメーターを表示してみましょう。 今回は ^mgstat によって提供されるパラメーターのみを使用します。 また、Docker コンテナについても説明のために少しだけ取り上げます。



Docker のインストール

最初のパートでは tarball から Prometheus と Grafana をインストールする方法を説明しています。 ここでは Docker の機能を使用して監視サーバーを起動する方法を説明します。 以下はデモ用のホストマシンです。

```
# uname -r
4.8.16-200.fc24.x86_64
# cat /etc/fedora-release
Fedora release 24 (Twenty Four)
```

さらに 2 台の仮想マシン（192.168.42.131 と 192.168.42.132）が VMWare Workstation Pro 12.0

環境で使用され、どちらも Caché がインストールされています。 これらのマシンが監視対象になります。
バージョンは次のとおりです。
```
 # uname -r
3.10.0-327.el7.x8664
# cat /etc/redhat-release
Red Hat Enterprise Linux Server release 7.2 (Maipo)
…
USER>write $zversion
Cache for UNIX (Red Hat Enterprise Linux for x86-64) 2016.2 (Build 721U) Wed Aug 17 2016 20:19:48 EDT
```

ホストマシンに Docker をインストールして起動しましょう。
```
 # dnf install -y docker
# systemctl start docker
# systemctl status docker
● docker.service — Docker Application Container Engine
Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
Active: active (running) since Wed 2017-06-21 15:08:28 EEST; 3 days ago
...
```

Docker コンテナで Prometheus を起動する

最新の Prometheus イメージをロードしましょう。
```
 # docker pull docker.io/prom/prometheus
```

Docker ファイルを参照すると、イメージが /etc/prometheus/prometheus.yml
ファイルから構成を読み取り、収集された統計が /prometheusフォルダーに保存されていることがわかります。
```
 …
CMD [ "-config.file=/etc/prometheus/prometheus.yml", \
"-storage.local.path=/prometheus", \
...
```

Docker コンテナで Prometheus
を起動する際に、ホストマシンから構成ファイルとメトリックデータベースをロードするようにしましょう。
こうすることで、コンテナの再起動を「乗り切る」ことができます。 次に、ホストマシン上に Prometheus
用のフォルダーを作成しましょう。
```
 # mkdir -p /opt/prometheus/data /opt/prometheus/etc
```

そして、次のような Prometheus の構成ファイルを作成しましょう。
```
 # cat /opt/prometheus/etc/prometheus.yml
global:
  scrapeinterval: 10s
scrapeconfigs:
  - jobname: 'isccache'
  metricspath: '/mgstat/5' #   Tail 5 (sec) it's a diff time for ^mgstat. Should be less than scrape interval.
  staticconfigs:
  - targets: ['192.168.42.131:57772','192.168.42.132:57772']
  basicauth:
  username: 'PromUser'
  password: 'Secret'
```

これで、Prometheus を含むコンテナを起動できます。
```
 # docker run -d --name prometheus \
--hostname prometheus -p 9090:9090 \
-v /opt/prometheus/etc/prometheus.yml:/etc/prometheus/prometheus.yml \
-v /opt/prometheus/data/:/prometheus \
docker.io/prom/prometheus
```

正常に起動したかどうかを確認してください。

```
 # docker ps --format "{{.ID}}: {{.Command}} {{.Status}} {{.Names}}"
d3a1db5dec1a: "/bin/prometheus -con" Up 5 minutes prometheus
```

Docker コンテナで Grafana を起動する

まずは最新のイメージをダウンロードしましょう。

```
 # docker pull docker.io/grafana/grafana
```

次に、Grafana データベース（デフォルトでは SQLite）をホストマシンに保存するように指定して起動します。
また、Prometheus を含むコンテナへのリンクを作成し、Grafana
を含むコンテナからそのコンテナにリンクできるようにします。

```
 # mkdir -p /opt/grafana/db
# docker run -d --name grafana \
--hostname grafana -p 3000:3000 \
--link prometheus \
-v /opt/grafana/db:/var/lib/grafana \
docker.io/grafana/grafana
# docker ps --format "{{.ID}}: {{.Command}} {{.Status}} {{.Names}}"
fe6941ce3d15: "/run.sh" Up 3 seconds grafana
d3a1db5dec1a: "/bin/prometheus -con" Up 14 minutes prometheus
```

Docker-compose を使用する

両方のコンテナは別々に起動されます。 Docker-compose
を使用すると、まとめて複数のコンテナを起動できるので便利です。 このツールをインストールし、現在の 2
つのコンテナを一時停止しましょう。その後、Docker-compose
経由で再起動するように再構成し、これらのコンテナをもう一度起動します。

これを cli で書くと次のようになります。

```
 # dnf install -y docker-compose
# docker stop $(docker ps -a -q)
# docker rm $(docker ps -a -q)
# mkdir /opt/docker-compose
# cat /opt/docker-compose/docker-compose.yml
version: '2'
services:
  prometheus:
  image: docker.io/prom/prometheus
  containername: prometheus
  hostname: prometheus
  ports:
  - 9090:9090
  volumes:
  - /opt/prometheus/etc/prometheus.yml:/etc/prometheus/prometheus.yml
  - /opt/prometheus/data/:/prometheus
  grafana:
  image: docker.io/grafana/grafana
  containername: grafana
  hostname: grafana
  ports:
  - 3000:3000
  volumes:
  - /opt/grafana/db:/var/lib/grafana
# docker-compose -f /opt/docker-compose/docker-compose.yml up -d
# # 両方のコンテナを次のコマンドを使用して無効化および削除できます。
# # docker-compose -f /opt/docker-compose/docker-compose.yml down

# docker ps --format "{{.ID}}: {{.Command}} {{.Status}} {{.Names}}"
620e3cb4a5c3: "/run.sh" Up 11 seconds grafana
```

e63416e6c247: "/bin/prometheus -con" Up 12 seconds prometheus

インストール後の手順

Grafana を初めて起動した後は、Web
インターフェイスの管理者パスワードを変更し（デフォルトでは、ログインとパスワードの組み合わせは
admin/admin です）、Prometheus をデータソースとして追加する必要があります。 この手順は Web
インターフェイスから実施できますが、Grafana SQLite データベース（デフォルトの位置は
/opt/grafana/db/grafana.db）を直接編集するか、REST リクエストを使用することによっても実施できます。

さらにもう 1 つの方法をご紹介します。次をご覧ください。
```
 # curl -XPUT "admin:admin@localhost:3000/api/user/password" \
-H "Content-Type:application/json" \
-d '{"oldPassword":"admin","newPassword":"TopSecret","confirmNew":"TopSecret"}'
```
パスワードが正常に変更された場合は次の応答が返ってきます。
{"message":"User password changed"}

次のような応答が返ってくる場合があります：
curl: (56) Recv failure: Connection reset by peer
これは、Grafana サーバーがまだ起動しておらず、前述のコマンドを再度実行する前に少し待機する必要があることを意味しています。 例えば、次のように待機できます。

```
# until curl -sf admin:admin@localhost:3000 > /dev/null; do sleep 1; echo "Grafana is not started yet";done; echo "Grafana is started"
```

パスワードを正常に変更したら、Prometheus のデータソースを追加してください。

```
# curl -XPOST "admin:TopSecret@localhost:3000/api/datasources" \
-H "Content-Type:application/json" \
-d '{"name":"Prometheus","type":"prometheus","url":"http://prometheus:9090","access":"proxy"}'
```

データソースが正常に追加されると、次の応答が返ってきます。
{"id":1,"message":"Datasource added","name":"Prometheus"}

^mgstat に相当するものを作成する

^mgstat は対話モードで出力をファイルとターミナルに保存します。
ここではファイルへの出力は取り上げません。 このため、Studio を使用して USER スペースに ^mgstat
をオブジェクト指向で実装した my.Metrics というクラスを作成してコンパイルします。

```
/// このクラスは ^mgstat ルーチンをオブジェクト指向で実装しています。
/// 前回とは異なり、Caché のバージョンチェックはスキップされます。
/// seizes を監視したい場合はパラメーター ISSEIZEGATHERED を 1 に設定する必要があります。
/// ^mgstat ルーチンとは異なり、Seizes メトリックは（パーセンテージではなく）差分として表示されます。
/// 一部の $zutil 関数についてはよく分かりませんが、^mgstat で使用されているので残しておきます。
Class my.Metrics Extends %RegisteredObject
{

/// メトリックの接頭辞
Parameter PREFIX = "isccachemgstat";

/// Prometheus のメトリックは改行で区切る必要があります。
Parameter NL As COSEXPRESSION = "$c(10)";

/// 不明なパラメーターです -) ^mgstat.int と同じものを使用しています。
Parameter MAXVALUE = 1000000000;
```

```
/// 2**64 - 10 です。 なぜマイナス 10 なのでしょうか？ 分かりません -) ^mgstat.int
と同じものを使用しています。
Parameter MAXVALGLO = 18446744073709551610;

/// 監視対象にするリソースです。 このリストは変更できます。
Parameter SEIZENAMES = "Global,ObjClass,Per-BDB";

/// デフォルト値は $zutil(69,74) です。 "1" を設定すると seize 統計の収集を開始できます。
Parameter ISSEIZEGATHERED = 0;

Parameter MAXECPCONN As COSEXPRESSION = "$system.ECP.MaxClientConnections()";

/// グローバルバッファタイプの数（8K、16K など）
Parameter NUMBUFF As COSEXPRESSION = "$zutil(190, 2)";

/// メモリオフセット（用途不明 ）
Parameter WDWCHECK As COSEXPRESSION = "$zutil(40, 2, 146)";

/// 書き込みデーモンフェーズ用のメモリオフセット
Parameter WDPHASEOFFSET As COSEXPRESSION = "$zutil(40, 2, 145)";

/// ジャーナル用のオフセット
Parameter JOURNALBASE As COSEXPRESSION = "$zutil(40, 2, 94)";

ClassMethod getSamples(delay As %Integer = 2) As %Status
{
  set sc = $$$OK
  try {
  set sc = ..gather(.oldValues)
  hang delay
  set sc = ..gather(.newValues)
  set sc = ..diff(delay, .oldValues, .newValues, .displayValues)
  set sc = ..output(.displayValues)
  } catch e {
  write "Error: "e.Name""e.Location, ..#NL
  }
  quit sc
}

ClassMethod gather(Output values) As %Status
{
  set sc = $$$OK

  // グローバルの統計を取得
  set sc = ..getGlobalStat(.values)

  // 書き込みデーモンの統計を取得
  set sc = ..getWDStat(.values)

  // ジャーナルの書き込みを取得
  set values("journalwrites") = ..getJournalWrites()

  // seize の統計を取得
  set sc = ..getSeizeStat(.values)

  // ECP の統計を取得
  set sc = ..getECPStat(.values)

  quit sc
```

```
}

ClassMethod diff(delay As %Integer = 2, ByRef oldValues, ByRef newValues, Output displayValues) As %Status
{
  set sc = $$$OK

  // グローバルのメトリックを処理
  set sc = ..loopGlobal("global", .oldValues, .newValues, delay, 1, .displayValues)

  set displayValues("readratio") = $select(
  displayValues("physicalreads") = 0: 0,
  1: $number(displayValues("logicalblockrequests") / displayValues("physicalreads"),2)
  )
  set displayValues("globalremoteratio") = $select(
  displayValues("remoteglobalrefs") = 0: 0,
  1: $number(displayValues("globalrefs") / displayValues("remoteglobalrefs"),2)
  )

  // 書き込みデーモンのメトリックを処理（秒単位ではない）
  set sc = ..loopGlobal("wd", .oldValues, .newValues, delay, 0, .displayValues)

  // ジャーナルの書き込みを処理
  set displayValues("journalwrites") = ..getDiff(oldValues("journalwrites"), newValues("journalwrites"), delay)

  // seize メトリックの処理
  set sc = ..loopGlobal("seize", .oldValues, .newValues, delay, 1, .displayValues)

  // ECP クライアントメトリックの処理
  set sc = ..loopGlobal("ecp", .oldValues, .newValues, delay, 1, .displayValues)
  set displayValues("actecp") = newValues("actecp")

  quit sc
}

ClassMethod getDiff(oldValue As %Integer, newValue As %Integer, delay As %Integer = 2) As %Integer
{
  if (newValue set diff = (..#MAXVALGLO - oldValue + newValue) \delay
  if (diff > ..#MAXVALUE) set diff = newValue \delay
  } else {
  set diff = (newValue - oldValue) \delay
  }
  quit diff
}

ClassMethod loopGlobal(subscript As %String, ByRef oldValues, ByRef newValues, delay As %Integer = 2,
perSecond As %Boolean = 1, Output displayValues) As %Status
{
  set sc = $$$OK

  set i = ""
  for {
  set i = $order(newValues(subscript, i))
  quit:(i = "")
  if (perSecond = 1) {
  set displayValues(i) = ..getDiff(oldValues(subscript, i), newValues(subscript, i), delay)
  } else {
  set displayValues(i) = newValues(subscript, i)
  }
  }
```

```
  quit sc
}

ClassMethod output(ByRef displayValues) As %Status
{
  set sc = $$$OK
  set i = ""
  for {
  set i = $order(displayValues(i))
  quit:(i = "")
  write ..#PREFIXi," ", displayValues(i),..#NL
  }
  write ..#NL
  quit sc
}

ClassMethod getGlobalStat(ByRef values) As %Status
{
  set sc = $$$OK

  set gloStatDesc = "routinerefs,remoteroutinerefs,routineloadsandsaves,"_
  "remoteroutineloadsandsaves,globalrefs,remoteglobalrefs,"_
  "logicalblockrequests,physicalreads,physicalwrites,"_
  "globalupdates,remoteglobalupdates,routinecommands,"_
  "wijwrites,routinecachemisses,objectcachehit,"_
  "objectcachemiss,objectcacheload,objectreferencesnewed,"_
  "objectreferencesdel,processprivateglobalrefs,processprivateglobalupdates"

  set gloStat = $zutil(190, 6, 1)

  for i = 1:1:$length(gloStat, ",") {
  set values("global", $piece(gloStatDesc, ",", i)) = $piece(gloStat, ",", i)
  }

  quit sc
}

ClassMethod getWDStat(ByRef values) As %Status
{
  set sc = $$$OK

  set tempWdQueue = 0
  for b = 1:1:..#NUMBUFF {
  set tempWdQueue = tempWdQueue + $piece($zutil(190, 2, b), ",", 10)
  }

  set wdInfo = $zutil(190, 13)
  set wdPass = $piece(wdInfo, ",")
  set wdQueueSize = $piece(wdInfo, ",", 2)
  set tempWdQueue = tempWdQueue - wdQueueSize
  if (tempWdQueue
  set misc = $zutil(190, 4)
  set ijuLock = $piece(misc, ",", 4)
  set ijuCount = $piece(misc, ",", 5)

  set wdPhase = 0
  if (($view(..#WDWCHECK, -2, 4)) && (..#WDPHASEOFFSET)) {
  set wdPhase = $view(..#WDPHASEOFFSET, -2, 4)
```

```
 }

 set wdStatDesc = "writedaemonqueuesize,writedaemontempqueue,"_
 "writedaemonpass,writedaemonphase,ijulock,ijucount"

 set wdStat = wdQueueSize","tempWdQueue","wdPass","wdPhase","ijuLock","ijuCount

 for i = 1:1:$length(wdStat, ",") {
 set values("wd", $piece(wdStatDesc, ",", i)) = $piece(wdStat, ",", i)
 }

 quit sc
}

ClassMethod getJournalWrites() As %String
{
 quit $view(..#JOURNALBASE, -2, 4)
}

ClassMethod getSeizeStat(ByRef values) As %Status
{
 set sc = $$$OK

 set seizeStat = "", seizeStatDescList = ""
 set selectedNames = ..#SEIZENAMES

 set seizeNumbers = ..getSeizeNumbers(selectedNames) // seize statistics
 set isSeizeGatherEnabled = ..#ISSEIZEGATHERED
 if (seizeNumbers = "") {
 set SeizeCount = 0
 } else {
 set SeizeCount = isSeizeGatherEnabled * $length(seizeNumbers, ",")
 }

 for i = 1:1:SeizeCount {
 set resource = $piece(seizeNumbers, ",", i)
 set resourceName = ..getSeizeLowerCaseName($piece(selectedNames, ",", i))
 set resourceStat = $zutil(162, 3, resource)
 set seizeStat = seizeStat$listbuild($piece(resourceStat, ","))
 set seizeStat = seizeStat$listbuild($piece(resourceStat, ",", 2))
 set seizeStat = seizeStat$listbuild($piece(resourceStat, ",", 3))
 set seizeStatDescList = seizeStatDescList$listbuild(
 resourceName"seizes", resourceName"nseizes", resourceName"aseizes"
 )
 }
 set seizeStatDesc = $listtostring(seizeStatDescList, ",")

 set seizeStat = $listtostring(seizeStat, ",")

 if (seizeStat '= "") {
 for k = 1:1:$length(seizeStat, ",") {
 set values("seize", $piece(seizeStatDesc, ",", k)) = $piece(seizeStat, ",", k)
 }
 }

 quit sc
}

ClassMethod getSeizeNumbers(selectedNames As %String) As %String
```

```
{
  /// USER>write $zu(162,0)
  // Pid,Routine,Lock,Global,Dirset,SatMap,Journal,Stat,GfileTab,Misc,LockDev,ObjClass...
  set allSeizeNames = $zutil(162,0)"," //すべてのリソース名を返す

  set seizeNumbers = ""
  for i = 1:1:$length(selectedNames, ",") {
  set resourceName = $piece(selectedNames,",",i)
  continue:(resourceName = "")||(resourceName = "Unused")
  set resourceNumber = $length($extract(allSeizeNames, 1, $find(allSeizeNames, resourceName)), ",") - 1
  continue:(resourceNumber = 0)
  if (seizeNumbers = "") {
  set seizeNumbers = resourceNumber
  } else {
  set seizeNumbers = seizeNumbers","resourceNumber
  }
  }
  quit seizeNumbers
}

ClassMethod getSeizeLowerCaseName(seizeName As %String) As %String
{
  quit $tr($zcvt(seizeName, "l"), "-", "")
}

ClassMethod getECPStat(ByRef values) As %Status
{
  set sc = $$$OK

  set ecpStat = ""

  if (..#MAXECPCONN '= 0) {
  set fullECPStat = $piece($system.ECP.GetProperty("ClientStats"), ",", 1, 21)
  set activeEcpConn = $system.ECP.NumClientConnections()
  set addBlocks = $piece(fullECPStat, ",", 2)
  set purgeBuffersByLocal = $piece(fullECPStat, ",", 6)
  set purgeBuffersByRemote = $piece(fullECPStat, ",", 7)
  set bytesSent = $piece(fullECPStat, ",", 19)
  set bytesReceived = $piece(fullECPStat, ",", 20)
  }
  set ecpStatDesc = "addblocks,purgebufferslocal,"_
  "purgeserverremote,bytessent,bytesreceived"

  set ecpStat = addBlocks","purgeBuffersByLocal","_
  purgeBuffersByRemote","bytesSent","bytesReceived

  if (ecpStat '= "") {
  for l = 1:1:$length(ecpStat, ",") {
  set values("ecp", $piece(ecpStatDesc, ",", l)) = $piece(ecpStat, ",", l)
  }
  set values("actecp") = activeEcpConn
  }

  quit sc
}

}
```

REST 経由で my.Metrics を呼び出すため、USER スペースにラッパークラスを作成しましょう。

```
Class my.Mgstat Extends %CSP.REST
{

XData UrlMap [ XMLNamespace = "http://www.intersystems.com/urlmap" ]
{




}

ClassMethod getMgstat(delay As %Integer = 2) As %Status
{
  // デフォルトでは 2 秒間隔で平均値を取得します
  quit ##class(my.Metrics).getSamples(delay)
}

}
```

リソース、ユーザー、Web アプリケーションを作成する

メトリックを提供するクラスが完成し、RESTful Web アプリケーションを作成できるようになりました。
最初の記事と同様に、この Web アプリケーションにリソースを割り当て、そのリソースを使用して Prometheus がメトリックを収集するユーザーを作成します。
作成したら、特定のデータベースにユーザー権限を付与しましょう。 最初の記事とは異なり、CACHESYS データベースに書き込むための権限（ loop+1^mymgstat *gmethod" エラーを回避するため）を追加し、%AdminManage リソースを使用できるようにしました（ gather+10^mymgstat *GetProperty,%SYSTEM.ECP" エラーを回避するため）。 192.168.42.131 と 192.168.42.132 の両方の仮想サーバーでこれらの手順を繰り返しましょう。 ただし、その前に作成した my.Metrics クラスと my.Mgstat クラスのコードを両方のサーバーの USER スペースにアップロードします（コードは GitHub で取得できます）。

具体的にはそれぞれの仮想サーバーで次の手順を実行します。

```
  # cd /tmp
# wget https://raw.githubusercontent.com/myardyas/prometheus/master/mgstat/cos/udl/Metrics.cls
# wget https://raw.githubusercontent.com/myardyas/prometheus/master/mgstat/cos/udl/Mgstat.cls
#
# # サーバーがインターネットに接続されていない場合はプログラムとクラスをローカル環境でコピーし、 scp を使用してください。
#
# csession -U user
USER>do $system.OBJ.Load("/tmp/Metrics.cls*/tmp/Mgstat.cls","ck")
USER>zn "%sys"
%SYS>write ##class(Security.Resources).Create("PromResource","Resource for Metrics web page","")
1
%SYS>write ##class(Security.Roles).Create("PromRole","Role for PromResource","PromResource:U,%AdminManage:U,%DBUSER:RW,%DBCACHESYS:RW")
1
%SYS>write ##class(Security.Users).Create("PromUser","PromRole","Secret")
1
%SYS>set properties("NameSpace") = "USER"
%SYS>set properties("Description") = "RESTfull web-interface for mgstat"
%SYS>set properties("AutheEnabled") = 32 ; 説明を参照してください
%SYS>set properties("Resource") = "PromResource"
%SYS>set properties("DispatchClass") = "my.Mgstat"
%SYS>write ##class(Security.Applications).Create("/mgstat",.properties)
```

1

curl を使用してメトリックにアクセスできることを確認する

（ファイアウォールで 57772 番ポートを忘れずに開いてください）
curl --user PromUser:Secret -XGET http://192.168.42.131:57772/mgstat/5
iscCacheMgstatGlobalRefs 347
iscCacheMgstatRemoteGlobalRefs 0
iscCacheMgstatGlobalRemoteRatio 0
…
curl --user PromUser:Secret -XGET http://192.168.42.132:57772/mgstat/5
iscCacheMgstatGlobalRefs 130
iscCacheMgstatRemoteGlobalRefs 0
iscCacheMgstatGlobalRemoteRatio 0
…

Prometheus からメトリックにアクセスできることを確認する

Prometheus は 9090 番ポートをリッスンします。 まずは [Targets] のステータスを確認しましょう。

その後、任意のメトリックを確認してください。

1 つのメトリックを表示する

ここでは 1 つのメトリック（iscCacheMgstatGlobalRefs）を例としてグラフに表示します。
まず、ダッシュボードを更新し、そこにグラフを挿入する必要があります。 そのためには
Grafana（http://localhost:3000、ログイン/パスワードは
admin/TopSecret）に移動し、新しいダッシュボードを追加してください。

グラフを追加します。

[Panel title]、[Edit] の順にクリックし、グラフを編集します。

Prometheus をデータソースとして設定し、iscCacheMgstatGlobalRefs メトリックを選択します。 解像度は
1/1 に設定します。

このグラフに名前を付けましょう。

凡例を追加します。

ウィンドウの上部にある [Save] ボタンをクリックし、ダッシュボードの名前を入力します。

最終的には次のように表示されます。

すべてのメトリックを表示する

残りのメトリックも同じように追加しましょう。 2 つのテキストメトリックがあります（Singlestat）。
その結果、次のダッシュボードが表示されます（ここでは上部と下部に分けて掲載しています）。

次の 2 つは明らかに問題があるように思われます。

— 凡例のスクロールバー（サーバーの数が増えるとスクロールバーが長くなります）。

— Singlestat パネルにデータがありません（値が単一であることを意味します）。 私たちには 2
台のサーバーとそれに対応する 2 つの値があります。

テンプレートを使用する

インスタンスにテンプレートを導入し、これらの問題を解決してみましょう。

そのためにはインスタンスの値を格納する変数を作成し、ルールに従って Prometheus へのリクエストを少しだけ編集する必要があります。 つまり、*'instance"* 変数を作成した後に "*isccachemgstatglobalrefs* " リクエストの代わりに"*isccachemgstatglobalrefs{instance="[[instance]]"}* " を使用する必要があります。

変数を作成します。

Prometheus へのリクエストでは、各メトリックからインスタンスラベルの値を選択しましょう。
画面の下のほうで 2 つのインスタンスの値が識別されていることがわかります。 [Add] ボタンをクリックしてください。

ダッシュボードの上部に、使用可能な値を持つ変数が追加されました。

次に、ダッシュボードの
各パネルのリクエストにこの変数を追加しましょう。つまり、"*isccachemgstatglobalrefs* "
のようなリクエストを "*isccachemgstatglobalrefs{instance="[[instance]]"}* " に変更します。
最終的なダッシュボードは次のようになります（インスタンス名は意図的に凡例の横に残されています）。

Singlestat パネルが機能するようになりました。

このダッシュボードのテンプレートは GitHub からダウンロードできます。 テンプレートを Grafana にインポートする手順はこの記事のパート 1で説明しています。

最後に、サーバー 192.168.42.132 を 192.168.42.131 の ECP クライアントにして ECP トラフィックを生成するためのグローバルを作成しましょう。 次のように、ECP クライアントの監視が機能していることがわかります。

まとめ

^mgstat の結果を Excel で表示する代わりに、見栄えの良いグラフを使ったダッシュボードをオンラインで使用することができます。
デメリットは、^mgstat の代替バージョンを使用しなければならないことです。
一般的には元になるツールのコードは変更される可能性がありますが、その事は考慮されていません。
ただし、非常に楽な方法で Caché のパフォーマンスを監視することができます。

最後までお読みいただき、ありがとうございました！

つづく...

追伸

デモ（1 つのインスタンス用）はこちらで利用できます。ログイン/パスワードは必要ありません。

#システム管理 #監視 #視覚化 #Caché

ソースURL:
https://jp.community.intersystems.com/post/grafana-%E3%83%99%E3%83%BC%E3%82%B9%E3%81%AE-mgstat%EF%BC%88intersystems-cach%C3%A9-ensemble-healthshare%E3%81%AE%E3%82%B7%E3%82%B9%E3%83%86%E3%83%A0%E7%9B%A3%E8%A6%96%E3%83%84%E3%83%BC%E3%83%AB%EF%BC%89%E7%94%A8-gui