記事
Toshihiko Minamoto · 2022年1月11日 3m read

データジャングルの視覚化 -- パート3: %SYS.MONLBLに基づくヒートマップの作成

これまでに何度もコードカバレッジとコードのパフォーマンス最適化
について説明してきたため、ほとんどの方はすでにSYS.MONLBLユーティリティについてご存知かと思います。
コードを視覚的に見る方が通常は、純粋な数値を見るよりもはるかに直感的に理解できます。これが、このシリーズの記事の大きなポイントです。 今回は、Pythonとそのツールから少し離れて、^%SYS.MONLBLレポートからヒートマップを生成する方法を探りたいと思います。

簡単に言うと、ヒートマップは特定の値を色で表現してデータの要約を得ることに特化した視覚化ツールです。
このケースでは、データはコード行であり、コード行に掛けられた時間が色にマッピングされます。

^%SYS.MONLBL

行ごとに監視するモニターの実行については、ドキュメントをご覧ください。
つまり、分析の完全な出力をCSVファイルとして操作します。
分析しようとしているコードのソースコードが実際にあれば、はるかに便利であるため、 *k*
フラグ（ソースを保持）を使ってコードをコンパイルするようにしてください。

出力の準備

ターゲット出力として、準備されたhtmlファイルを使用することにします。
これには、非常に基本的なレイアウトと、最終的な色付けを行うための小さなJavaScript関数だけが含まれます。

```
<!doctype html>
<html class="no-js" lang="">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="x-ua-compatible" content="ie=edge">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <!-- Place favicon.ico and apple-touch-icon.png in the root directory -->
        <link rel="apple-touch-icon" href="apple-touch-icon.png">

        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize
/4.2.0/normalize.min.css">

        <!--<link rel="stylesheet" href="css/main.css"> -->
            <style>

        table, th, td {
            width:"100%";
            border: 1px solid black;
            border-collapse: collapse;
        }
```

```
        pre {
            margin:1px;
        }
        th {
            text-align: left;
        }
    </style>


    </head>

    <body onload="colorize()">
        <!--[if lt IE 8]>
            <p class="browserupgrade">You are using an <strong>outdated</strong> brow
ser. Please <a href="http://browsehappy.com/">upgrade your browser</a> to improve you
r experience.</p>
        <![endif]-->




<table id="data">
<tr><th>Routine</th><th>Line</th><th>Total Time</th><th>Code</th></tr>
<!--output-->
</table>


    </body>
</html>
```

解析してまとめる

生成されたCSVから関連する情報を取得してテンプレートに入れるには、以下のスクリプトを使用します。

monlbl.sh

```
#!/bin/bash

cat $1|grep -vi totals| awk -F"," 'FNR>1 {out="<tr><td>"$1"</td>" "<td>" $2 "</td><td
>" $54 "</td><td><pre>"; for(i=55;i<=NF;i++){out=out$i","}; out=substr(out, 1, length
(out)-1) "</pre></td></tr>"; print out }'
```

gen-heatmap.sh

```
#!/bin/bash
./monlbl.sh $1 > /tmp/temp.data
sed -e '/<!--output-->/r/tmp/temp.data' template.html
```

上記を以下のようにして呼び出します。

```
./gen-heatmap.sh /tmp/report.csv > heatmap.html
```

最終的な出力

%Library.PopulateUtils.1	126	0	" ""Uma"",""Usha"",""Valery"",""Violet"",""Wilma"",""Yan"",""Zelda"",""Zoe"")"
%Library.PopulateUtils.1	127	0	" }"
%Library.PopulateUtils.1	128	0.778136	" Quit $LI(list,($zu(165,1,$LL(list)))+1)"
%Library.PopulateUtils.1	129	0	"zFloat(min=0,max=10000,scale=0) public {"
%Library.PopulateUtils.1	130	0	" If max '> min { Quit """" }"
%Library.PopulateUtils.1	131	0	" If scale > 17 { Set scale = 17 }"
%Library.PopulateUtils.1	132	0	" Set factor = 10 ** scale"
%Library.PopulateUtils.1	133	0	" Set minn = (min*factor\1)/factor, maxn = (max*factor\1)/factor"
%Library.PopulateUtils.1	134	0	" Set range = maxn - minn"
%Library.PopulateUtils.1	135	0	" If ($Length(range\1) + scale) < 17 { Quit +(((minn * factor) + ($zu(165,1,range * factor + 1))) / factor) }"
%Library.PopulateUtils.1	136	0	" set range = $Extract(range,1,17), factor = $Extract(factor,1,17)"
%Library.PopulateUtils.1	137	0	" Set float = minn + ($zu(165,1,range)) + $Select(scale:+(($zu(165,1,factor+1))/factor),1:0)"
%Library.PopulateUtils.1	138	0	" Quit $Select((floatmin):float,float'"
%Library.PopulateUtils.1	139	0	"zInteger(min=0,max=1000000000) public {"
%Library.PopulateUtils.1	140	0.744944	" if min > max quit 0"
%Library.PopulateUtils.1	141	0.168382	" set range = max - min + 1"
%Library.PopulateUtils.1	142	0.587527	" if range '> 1E17 { quit min+($zu(165,1,range)) }"
%Library.PopulateUtils.1	143	0	" set range1 = ($zu(165,1,1E17))"
%Library.PopulateUtils.1	144	0	" set range2 = ($zu(165,1,range\1E17))"
%Library.PopulateUtils.1	145	0	" quit min + (range2*1E17) + range1 }"
%Library.PopulateUtils.1	146	0	"zLastName()"

調整可能な要素

テンプレートに含まれる色付けの関数をよく見ると、時間には線形マッピングを使用していないことがわかります。

```
    function colorize() {
        var rows=$("#data tr")
        var max=Math.max.apply(Math,rows.slice(1,rows.length).map(function(){ return
this.childNodes[2].textContent}))
        for (i=1;i<rows.length;i++){
            var val=rows[i].childNodes[2].textContent;
            var c=(Math.pow(1-val/max,3))*255;
            var col=rgba(255,c,c,0.7);
            console.log(col);
            rows[i].style.backgroundColor=col;
        }
    }
```

私がテストした例では、この方が非常にうまく機能することがわかりましたが、その度合いは条件によって異なる可能性があります。 明らかに、指数を増やしてより赤に押し込むこともできますし、その逆も可能です。

コード

関連ファイルはすべてこちらにあります。

#ツール #パフォーマンス #視覚化 #Caché

ソースURL:
https://jp.community.intersystems.com/post/%E3%83%87%E3%83%BC%E3%82%BF%E3%82%B8%E3%83%A3%E3%83%B3%E3%82%B0%E3%83%AB%E3%81%AE%E8%A6%96%E8%A6%9A%E5%8C%96-%E3%83%91%E3%83%BC%E3%83%883-sysmonlbl%E3%81%AB%E5%9F%BA%E3%81%A5%E3%81%8F%E3%83%92%E3%83%BC%E3%83%88%E3%83%9E%E3%83%83%E3%83%97%E3%81%AE%E4%BD%9C%E6%88%90