课程实践答辩——VFG 可视化

孙子平

清华大学

2019 年 12 月 26 日

目录

项目总体介绍

目标

将 VFG（Value Flow Graph）**可视化出来**，供大家调试。

VFG 是用于程序静态分析的图。其节点表示程序运行产生的值。下图就是一个 VFG 的示例。

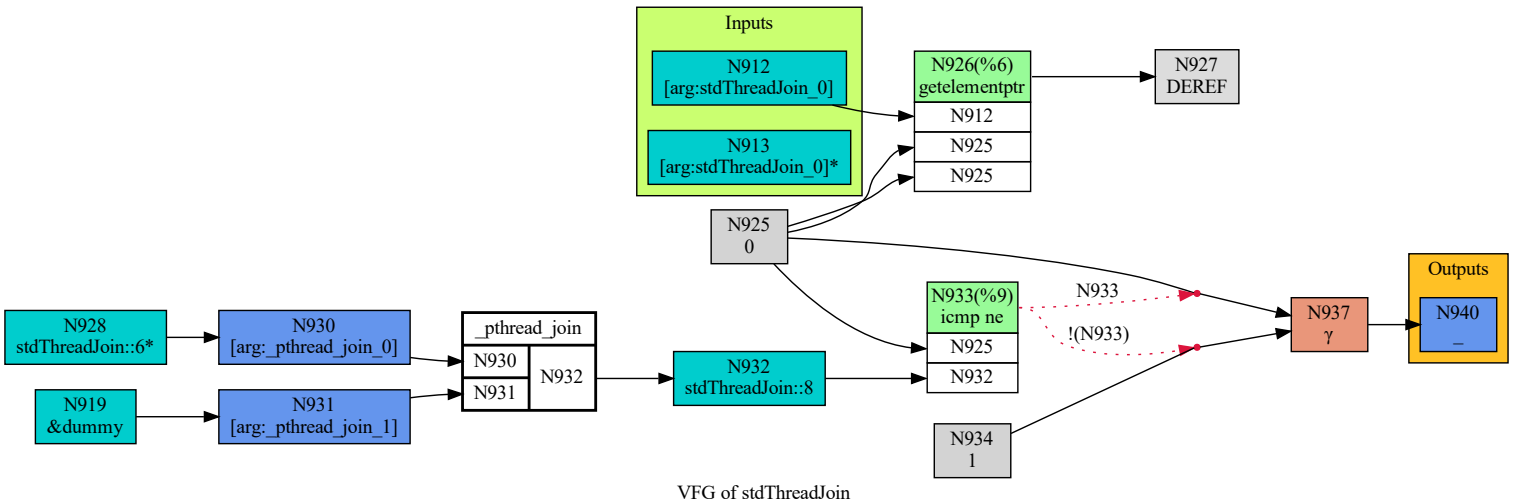

图 1: 由 Graphviz 可视化出的 VFG

Graphviz 简介

VFG 可以被导出成 Graphviz 文件（后缀名为.dot）。因而我的可视化软件的输入是 Graphviz 文件。下方是一段示例的 Graphviz 代码及其对应的图。Graphviz 文件有以下特点：

- 有三类元素：图、节点、边，可以有子图
- 图、节点和边都有自己的属性，如颜色、形状、标签
- 节点有类似 HTML 的表格节点和 Record 节点（之后会给示例）
- Graphviz 文件可以转成 Xdot 文件，与 Graphviz 文件具有相同文法，但多了作图属性方便其他软件绘制

```
digraph example {
  hello [color="red"]
  hello -> world
}
```

代码 1: Graphviz 示例代码

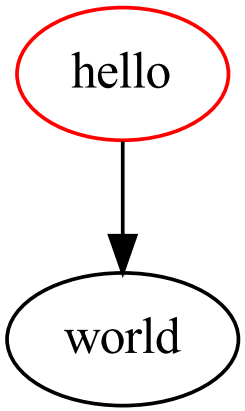

图 2: 左侧示例代码对应的图

为什么要重新写一个而不是用 Graphviz？

1. 我们需要交互效果，比如选中节点可以高亮它
2. 我们希望能够夸平台查看
3. 我们希望通过在数据中插入我们独有的属性，来表示出额外的信息

技术栈选择

技术	可选方案	选用	理由
呈现方式	网页、桌面	网页	跨平台
前端框架		Vue	Vue 很成熟、便捷
编程语言		JS，后改为 TS	TS 极大地减少了 bug
绘图方法	SVG、Canvas	Canvas	性能
Canvas 框架		Vue Konva，后改为手绘	性能

图 3: 技术栈的选择

可视化工具 UI

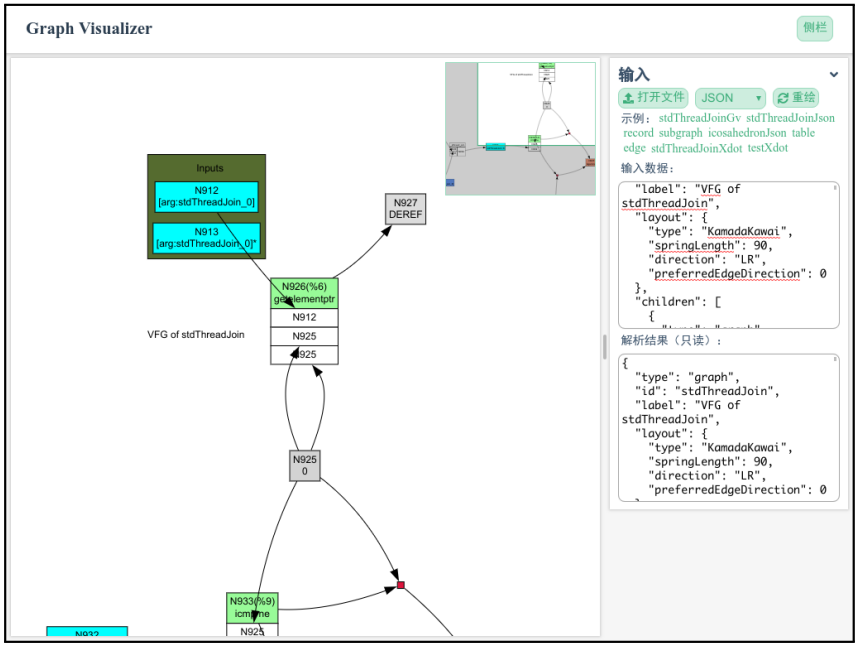

图 4: 可视化工具界面

可视化工具 UI 简介：

- 支持上传 Graphviz、Xdot 和 JSON 三种格式的文件
- 侧栏适配移动端，可收起或者改变宽度
- 有缩略图，可拖动缩略图的视框
- 左键拖动图或元素，滚轮可缩放图
- 对于输入可以增量更新

目录

类的结构

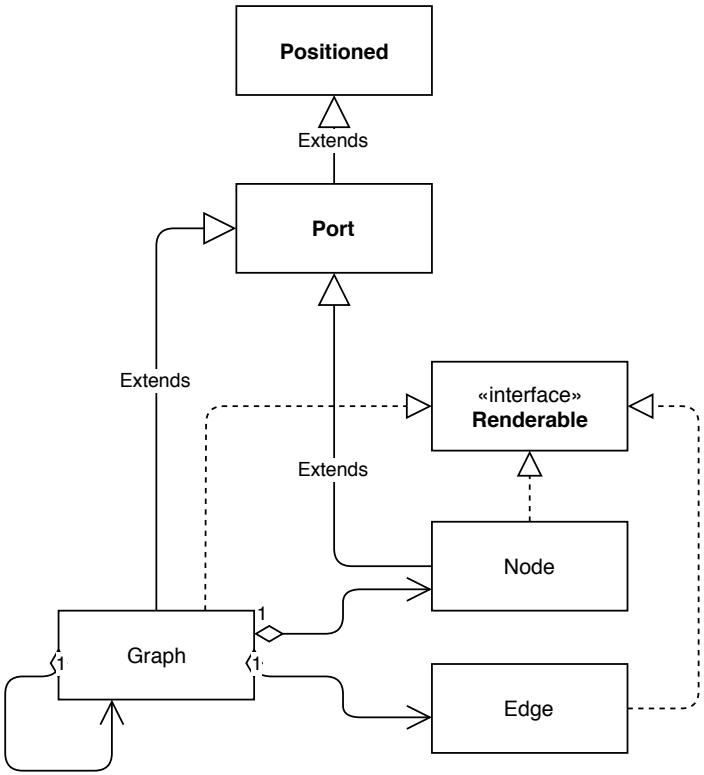

图 5: 可视化工具的主要类图结构

- 项目总代码破 6K 行，有着庞大的类关系，这里只显示主要类
- `Positioned` 表示有位置的，`Port` 表示可被边连接的
- 不包含在图中的类：
  - 每个实体 `Renderable` 类都有自己的形状子对象。如边有二次曲线、直线等，节点有方形、表格和 Record 节点
  - 对于边，它的控制点继承自 `Positioned`
  - 对于图，每个图的初始布局、物理布局、连通量布局算法都是一个类

目录

Graphviz 文件 Parser

```
graph       : [ strict ] (graph | digraph)
                [ ID ] '{' stmt_list '}'
stmt_list   : [ stmt [ ';' ] stmt_list ]
stmt        : node_stmt
            | edge_stmt
            | attr_stmt
            | ID '=' ID
            | subgraph
attr_stmt   : (graph | node | edge) attr_list
attr_list   : '[' [ a_list ] ']' [ attr_list ]
a_list      : ID '=' ID [ (';' | ',') ]
                [ a_list ]
edge_stmt   : (node_id | subgraph) edgeRHS
                [ attr_list ]
edgeRHS     : edgeop (node_id | subgraph)
                [ edgeRHS ]
node_stmt   : node_id [ attr_list ]
node_id     : ID [ port ]
port        : ':' ID [ ':' compass_pt ]
            | ':' compass_pt
subgraph    : [ subgraph [ ID ] ]
                '{' stmt_list '}'
compass_pt  : (n | ne | e | se | s | sw | w | nw
                | c | _)
```

代码 2: Graphviz 文法

- 为了将 Graphviz 解析出来供绘图使用，我依照 Graphviz 的文法写了个 Tokenizer 和 Parser
- 此外对于 Xdot 的作图属性，我也写了个简单的解析器，其文法属于正则文法，这里不再给出
- 部分代码借鉴了 `jrfonseca/xdot.py`，并提出其代码的 2 个 bug

Graphviz 绘图

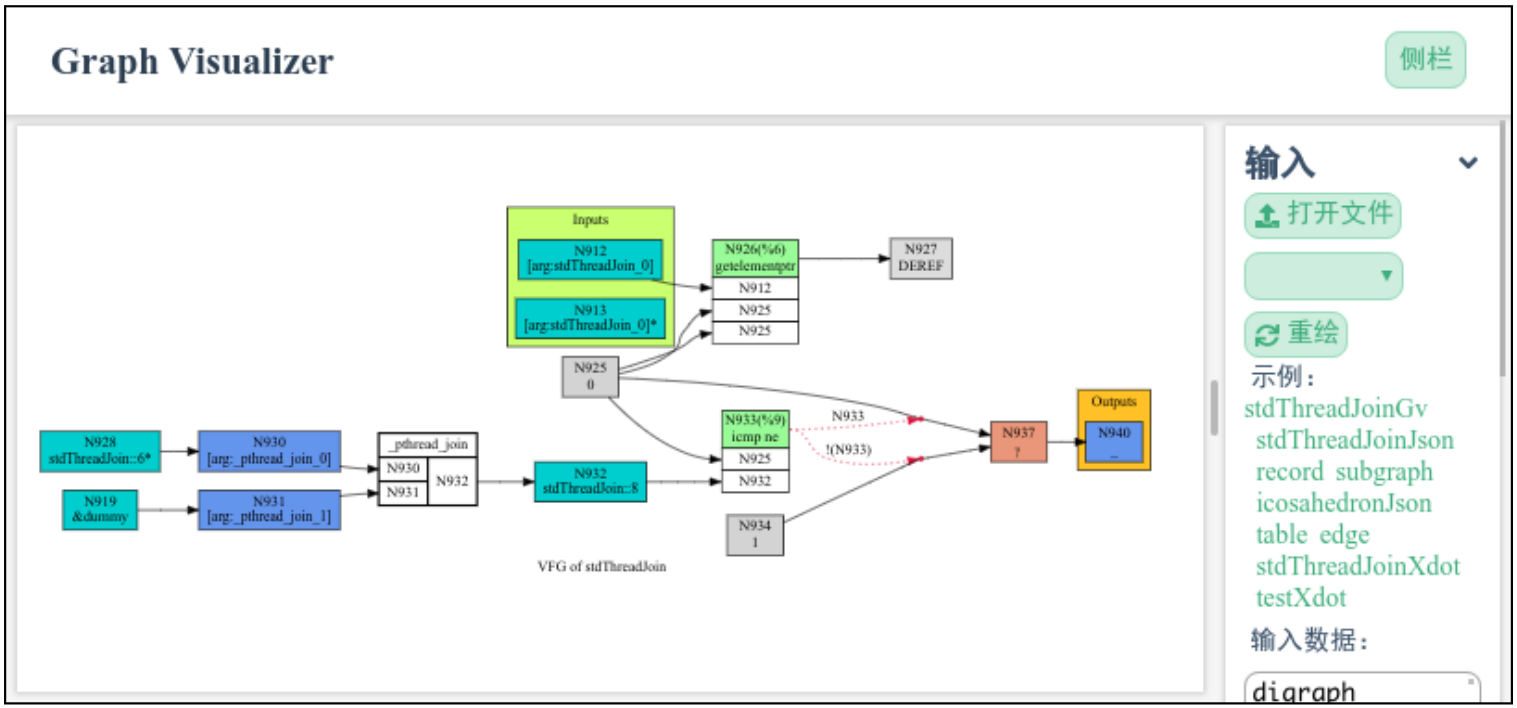

图 6: Xdot 文件的可视化结果

- Xdot 文件的可视化很好地还原了 Graphviz
- 由于 Xdot 文件没有详细指明绘图的节点与边的关系，且 Graphviz 的布局算法未知，所以较难实现拖拽元素，交互效果较差

目录

初始布局引擎——KamadaKawai

- KamadaKawai 是个经典的无向图布局算法，它假设无向图是个弹簧系统，并通过牛顿迭代法求解系统最低能量
- 我重新实现了这个算法，并且对于系统能量采用了增量更新以减少计算
- 部分代码借鉴了 `visjs/vis-network`，并提出其代码的 6 个 bug

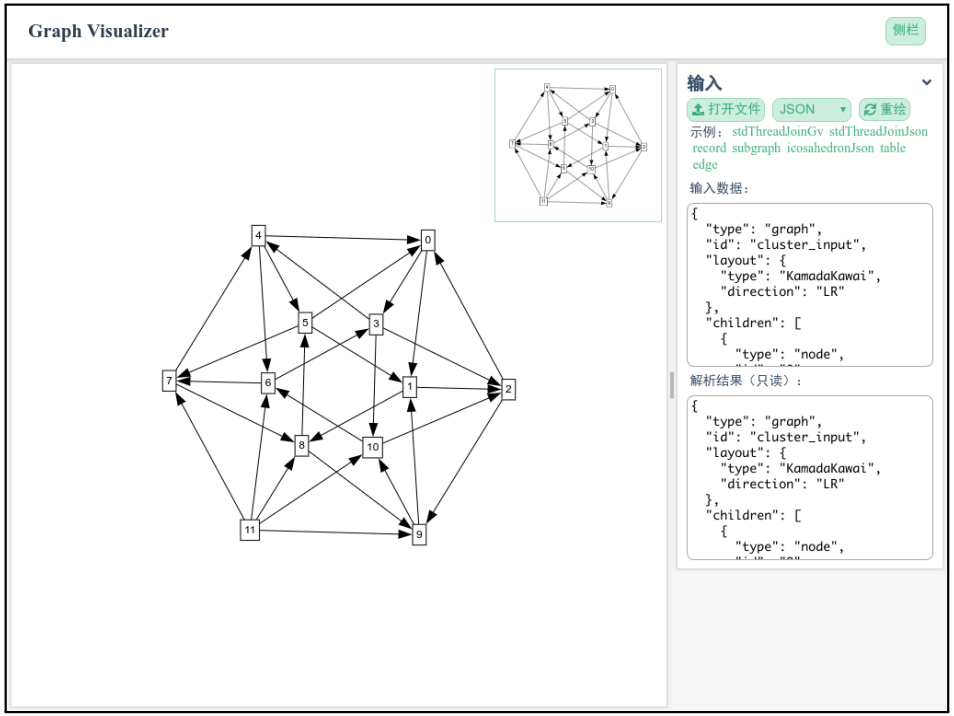

图 7: **直边**下 KamadaKawai 初始布局效果

物理布局引擎

- 有了物理布局引擎，就可以在初始布局的情况下继续更好地修正，并且更重要的是，它能使被拖动节点周围的边和节点一起动，并且使线弯曲，有更好的交互效果
- 我主要采用了以下 3 种力，此外还有摩擦力：

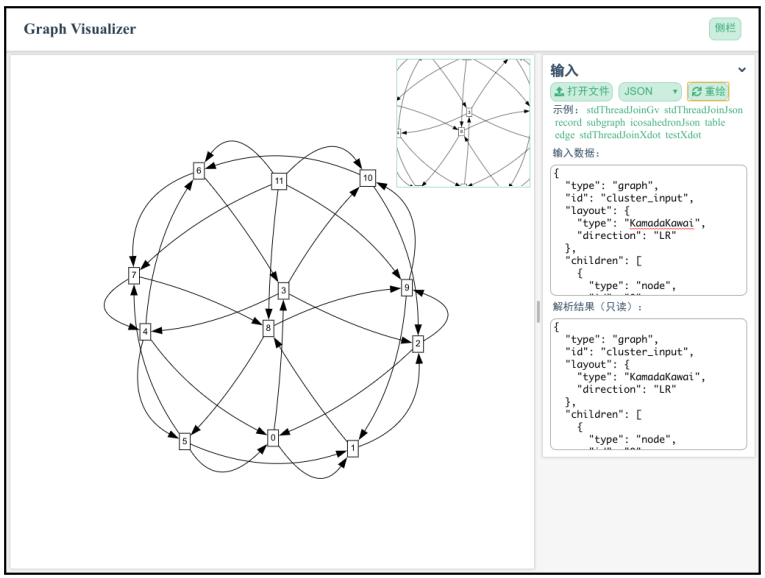

图 8: **曲边**下物理布局效果

- 弹簧力：每个边的控制点和端之间有类似弹簧的力
- 斥力：每个节点和控制点之间有斥力，斥力的计算采用了 BarnesHut 算法，类似 k-d 树，可以将复杂度降为 $O(n \log n)$
- 中心引力：会将所有的节点和控制点拉向中间

Record 节点

Record 节点可以递归地创建列表嵌套列表的结果，子列表和父列表的排列方向是水平竖直交替的。一个示例的 record 标签如下：

```
hello\nworld |{ b |{c|<here> d|e}| f}| g | h
```

我写了 Record 节点标签的解析器，并成功可视化了 Record 节点

```
recordLabel
  : field ('|' field )*
  ;
field
  : (' <'  string ' >' )? string?
  | '{' recordLabel '}'
```

代码 3: Record 节点标签的文法

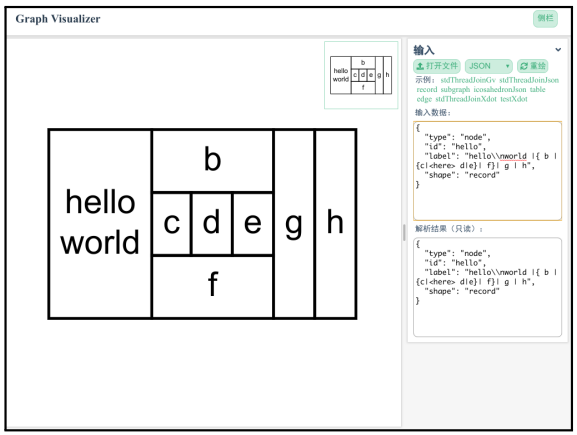

图 9: Record 节点可视化效果

Table 节点

Graphviz 还支持包含 HTML 元素 `table` 的节点。我调用了一个 XML 解析库得到了其语法树。由于网上搜不到表格的排版算法，我就自己造了一个。

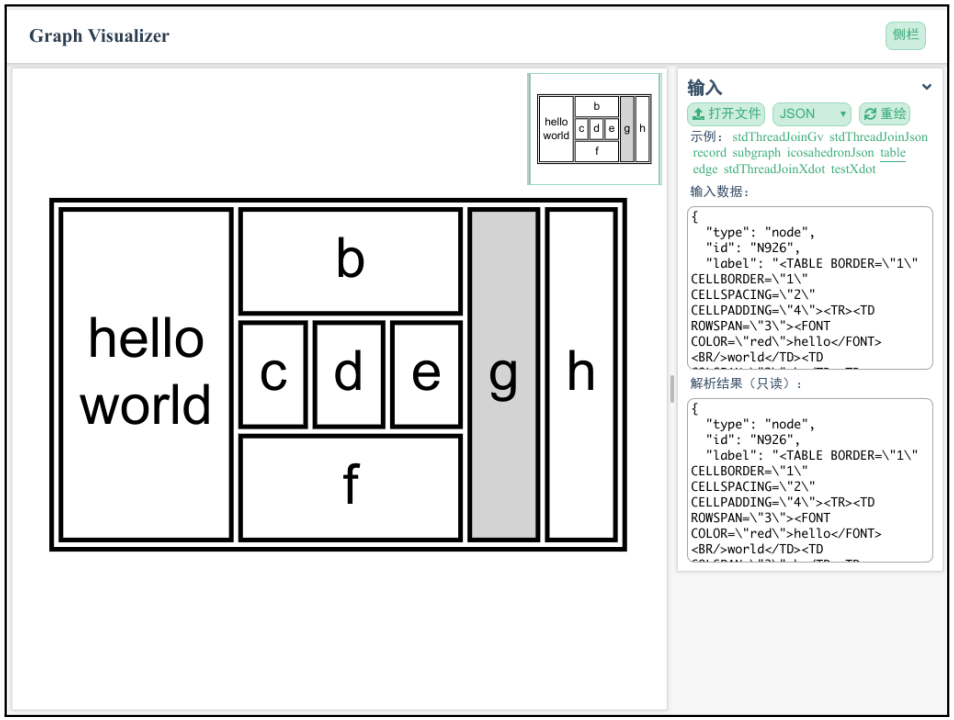

图 10: Table 节点可视化效果

目录

结论

总的而言，我使用两种方法可视化 VFG 导出的 Graphviz 文件，这两种方法优缺点如下：

- Xdot：优点实现简单、布局效果好，缺点需要 Graphviz 转换、交互效果好
- 非 Xdot：优点交互效果好、无向图布局好，缺点 VFG 布局不好