Advent of Code 2016 Day22: Grid Computing

Danny Wijnschenk · Nov 22, 2017

#Caché #Code Snippet #Contest #ObjectScript

This is a series of programming challenges for beginners and experienced Caché programmers.

For an introduction : go to article https://community.intersystems.com/post/advent-code-2016-day1-no-time-ta...

The goal of today's challenge is to copy data from one disk to another, problem is that the disks are not directly connected. Luckily they are in a huge grid, and you have to find a way to copy the data through adjacents disks, while honoring the disk usage (you cannot copy to a disk where data is already stored).

The input is a df-like disk usage listing.

The first challenge is to calculate a list of disks that can copy data.

These are all viable pairs of disk nodes (A,B) where:

```
- Node A is not empty (its Used is not zero).
- Nodes A and B are not the same node.
- The data on node A (its Used) would fit on node B (its Avail).
```

How many viable pairs of nodes are there?

The complete explanation of the challenge is here : http://adventofcode.com/2016/day/22, which also links to your puzzle input (df listing).

Here is the code i wrote, which was not that difficult :

```
Class AOC2016.Day22 Extends AOC2016.Utils
{

ClassMethod Part1(file As %String = "day22.txt")
{
  #Dim input, df, node, nodeA, nodeB as %String
  #Dim iInput, viable as %Integer
  #Dim nodes as Array of %String

  Try {
    Do ..Input(file, .input)
    For iInput=3:1:input {  //first two lines are df header
      Set df=$ZStrip(input(iInput),"<=>W")
      Set node=$Piece(df," ",1)
      Set nodes(node)=$ListFromString($Piece(df," ",2,*)," ")
    }
    Set viable = 0
```

```
    Set nodeA="" For  {
      Set nodeA = $Order(nodes(nodeA)) If nodeA="" Quit
      If +$List(nodes(nodeA),2)'=0 {  //A is not empty
        Set nodeB="" For {
          Set nodeB = $Order(nodes(nodeB)) If nodeB="" Quit
          If nodeA'=nodeB {  //A and B can not be the same node
            If +$List(nodes(nodeA),2) '> +$List(nodes(nodeB),3)
{  //used on A <= avail on B
              set viable=viable+1
            }
          }
        }
      }
    }
    Write "Viable nodes : ",viable,!
  } catch {
    Write "Error : ",$ZError,!
  }
}

}
```

The second part of the challenge is to start copying data from the upper right node in the grid to the upper left one, by always using one disk that is empty. There are also some disks that are huge, and cannot be used to copy to.

The whole puzzle can be seen as a big sliding puzzle, like the one where you have a frame of square tiles and one tile missing. (https://en.wikipedia.org/wiki/15_puzzle) (and with a little snag since there are some nodes you cannot use)

The challenge is to find the fewest number of steps required to move the data.

I broke down the challenge in two parts :

- find the 'empty' disk, and move it one-disk-left of the right upper corner.
- use the empty disk to move data of the disk to the left : in a repeating pattern, this will take 5 steps for every move to the left (look at the example on http://adventofcode.com/2016/day/22 to understand better)

So in total 5 times the amount of nodes on one row - 1.

The total steps is the sum of the two parts. (I know, it is a bit lazy not to write code for the second part, but I invite YOU to do that !)

So, here is my code :

```
ClassMethod Part2(file As %String = "day22.txt")
{
  #Dim input, df, node as %String
  #Dim iInput, x, y, maxX, maxY, freeX, freeY, minPath, pathLength as %Integer
  #Dim nodes as Array of %String
  set %path=""
  Try {
    Set (maxX,maxY,freeX,freeY)=0
    Do ..Input(file, .input)
    For iInput=3:1:input {  //first two lines df header
      Set df=$ZSTrip(input(iInput),"<=>W")
```

```
      Set node=$Piece(df," ",1)
      Set x = $Extract($Piece(node,"-",2),2,*)
      Set y = $Extract($Piece(node,"-",3),2,*)
      Set nodes(x,y)=$ListFromString($Piece(df," ",2,*)," ")
      If +$List(nodes(x,y),2)=0 set freeX=x, freeY=y
      If x>maxX Set maxX=x
      If y>maxY Set maxY=y
    }
    Set minPath=""
    Do ..Path(.nodes,freeX,freeY,maxX-1,0,"",.minPath) //path from free to before G
    Set pathLength=minPath+1+(5*(maxX-1))
    Do ..Output(.nodes,maxX,maxY)
    Write !,"minimum path : ",pathLength,!
  } catch {
    Write "Error : ",$ZError,!
  }
}

ClassMethod Path(nodes, x0, y0, x1, y1, path, minPath)
{
  #Dim xDelta, yDelta as %Integer
  #Dim moves as %String
  if (x0=x1) & (y0=y1) {
    If ($ListLength(path)<minPath)!(minPath="") Set minPath = $LL(path) w !,
minPath set %path=path
  } elseIf (($ListLength(path)+..Distance(x0,y0,x1,y1))'<minPath)&(minPath'="") {
    //to far from goal, even in a straight line : quit
  } else {
    For moves="0,-1","0,1","1,0","-1,0" {
      Set xDelta=$P(moves,",",1)
      Set yDelta=$P(moves,",",2)
      If '$Data(nodes(x0+xDelta,y0+yDelta)) Continue
      If $List(nodes(x0+xDelta,y0+yDelta),1)>100
Continue  //disk is too big to copy to
      If $ListFind(path,$lb(x0+xDelta,y0+yDelta)) Continue
      Do ..Path(.nodes,x0+xDelta,y0+yDelta,x1,y1,path_$lb($lb(x0+xDelta,y0+
yDelta)),.minPath)
    }
  }
}

///'Manhattan' distance : straight line
ClassMethod Distance(x0, y0, x1, y1)
{
  Quit $ZAbs(x1-x0)+$ZAbs(y1-y0)
}

ClassMethod Output(ByRef nodes, maxX, maxY)
{
  For y=0:1:maxY {
    For x=0:1:maxX {
      Write $Select(+$List(nodes(x,y),2)=0:"_",+$List(nodes(x,y),1)>100:"#",
$Find(%path,$lb(x,y)):"o",1:".")
    }
    Write !
  }
}
```

Look here for all our solutions so far : https://bitbucket.org/bertsarens/advent2016 and https://github.com/DannyWijnschenk/AdventOfCode2016

Here is the list of all Advent of Code 2016 articles :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

- 10
- 0
-
- 217
- 0

Reply

Source URL: https://community.intersystems.com/post/advent-code-2016-day22-grid-computing