WebIndex

Dealing with HUGE files

18 Aprile 2013

Section 1

Problem

Requisites

Webindex is a service to help a Client navigate a Multidimensional Model

- it ease the burden for the client to have all the model in memory
  - the model behaves like a open world video game

To do this, Webindex has to deconstruct a Huge volume into more memory (and network) friendly chunks

- But the Model can be too big to be analyzed in memory in the first place, it would be nice if the partitioner could analyze the Model in smaller chunks
  - chicken-egg problem

What is Huge?

```
Informatica Biomedica$ ls -hal tb278_800.vol
-rw-r--r-- 1 andrea andrea 626M apr 13  2003 tb278_800.vol
```

The Visible Human Project:

65 GB

Section 2

Solution

Indirect Access

Two basic operation are requested to partition a space:

- Provide an (unordered) iteration over Vertices of the Model
- Put a Vertex in a (unordered) Set

My sub-project is an **opaque layer** that gives the client program the impression that the Model's file is in Memory

Segmentation and buffers

or: What the Client library doesn't see

- **buffers** are setted up to map different blocks of the Model at different Times
  - i take advantage of the unordered iteration to minimize buffer's swaps
  - obviously more memory is better, as we are trading off memory size with access time
- the only Memory-Resident data structure is an **index** that keeps track of where on disk a vertex resides, and with which unordered set it belongs
  - the space cost for such operation could be: index of the block + offset within this block + python data structure's overhead.
  - But taking advantage of the locality principle, i can squash multiple successive vertices in the same structure, thus the cost to index a **sequence** of vertices is block + offset within this block + number of subsequent elements.
    - this is akin to what happens in a png encoded image

Why not Virtual Memory?

The solution proposed here is similar to what Virtual Memory does in a **OS**. Why reinvent the wheel?

VM is a **GENERIC** solution to a **GENERIC** problem, a custom solution can take advantage of the Information about the problem's Domain to be more efficient and to not clog the system to death.