AppRes (Apportion Resources)

**Introduction**

A personal computer is normally integrated with a core-2 dual processor and if we take the latest multiprocessing units we can find i3, i5, i7 processors (which are hyper-threading enabled).

It's not always that a working computer will be doing hyper-threading. Thus we have a resource that is in idle state i.e not been used by for any purpose. Thus the idea is to utilize all such idle processors of a system of computers connected via a network like: LAN network of a college where labs computers are connected, Companies where computers are connected through a network.

Idea was inspired after seeing lots of computer in our college lab unused or almost doing no work but are consuming power thus their resources which are actually very important are just useless since are not doing any work and are idle with CPU consuming power for no reason. On the other hand we have problems of memory storage, less of computation resources, extra computers to handle request etc.

Thus an idea has struck our minds as to what if we are able to connect such computers all over the local area network controlled by single by pass proxy server which can manage idle processors and can use the connected systems memory for some purpose thus providing a helping hand to the institute network administration.

Later a similar idea which was applied in companies local network idea proved to be good as during night time when huge number of computers are free inside the network they can over take the charge from main computers and the main computers can be maintained during that time thus without using extra resources we can keep the companies services up and running all the time and at the same time utilizing the existing resources in a better way or we can put these unused resources to some task thereby reducing the load on main servers.

Tasks like:
- Indexing,
- Data crunching,
- Handling request on web server,
- Doing computation intensive tasks.

Development of idea then took concepts from cloud-computing, parallel
computing, distributed computing and cryptography.

Problem Statement

To develop a software by means of which computation intensive tasks can be done efficiently without procuring any additional hardware on systems which are connected. The software manages the idle resources of a system of computers and ensures that

1. Memory: A portion of the available memory of each system is used thereby pooling up a large amount of space which otherwise would have been unused.

2. Processors: A task can be submitted to a processor, which is idle on any one of the connected system.

Design

Design can be viewed in terms of a Server Client architecture:

Server: Is the proxy server that will be the master and controlling all the processor and memory management.

Client: Systems connected to the server.

1.) Layout
A web proxy server through which all the network requests to outside world pass. And all the clients need to connect to it in order to connect to company / institution network as well as in order to connect to outside world.

Client
When a newly connected computer opens the browser for the first time, it will be redirected to a page, which will ask the user’s permission to join the software and the requirements. If the user wishes to share his computer resources, the amount of system memory he wishes to share and other relevant details are asked.

I. Configuring the Client System
Example: google.com -> (redirects to) AppRes main page and after filling up the details -> redirected back to google.com
Client visiting a site ( google.com )
Information required:

- Amount of Memory shared with a minimum of 1024MB physical memory.
• How AppRes will be using the system.
• If the client wishes to impose any restrictions on his memory access.

After filling all the details Client is automatically redirected to site (google.com)

II. Setting Up Client and Server Connection

Once client accepts the AppRes terms, a script is uploaded on clients computer (depending on the OS which was detected from browser). Since there can be issues due to firewall and permissions issues so a demo function is run so to check for all the memory access permission, processors state etc.

```c
Demo_Func() {
    sm = get_total_system_memory();
    no_core = get_no_of_cores();
    state = get_processors_state();
    permisson = make_file_folder();

    if no errors
        return { sm, no_core, state, permisson … } //to server
    else
        print alert on Client system
}
```

On proper execution of function, the client would send the report to the server while on error generation, errors along with solution are proposed to Client, after reconfiguring the system demo function is re-run to submit the report.

Unless Server get the confirmation from the demo function Client is not marked as a functional system in AppRes community [1]

Server

i. It contains Data Base: for storing a task table[2], complete task table[3], Total Available Memory (TAM) [4] etc.

ii. For keeping the copy of task and its task-table, we store certain flags which helps us understand the state of the system like:

Platform details: which Os / Memory Shared

Offline or Online (Active or Inactive or Non-active):

• Offline: System is not connected to the network.

• Online: System is connected to the network and is up and running might be in active or inactive state (as described below)
- **Active**: Client system currently running a task (Task is a work submitted by server to client it may include some job like: indexing, data crunching, handling request, storing temp files, acting like a cache server etc ...)

- **Inactive**: Client system is having the resources in idle state and can be utilized for doing some task.

- **Non-active**: Client system is online but is not having sufficient Total Free Memory (TFM)[7] or idle processor for task to take place. (In other words, client is not in a state to take a task from server)

iii. Script (a small code containing the demo function and center other functions that helps in communication between Server and Client): It also detects if the performance of other tasks running on the client have been affected because of the resource sharing.

2.) Implementation

**Q.) How is Server knowing that Client is in InActive / Non-Active state?**

**Approach:**
- The server can send periodic messages to the clients and wait for the acknowledgment for sometime. If there is no response from the Client, it can assume the Client is Offline. Since sending messages across a network isn't expensive this approach would be efficient.

- [OR] Instead of sending periodic messages client can send a message to the Server when ever it is ready for taking a task (thus server mark this client as an Inactive one and can be used later) and a message when its in Non-active state (thus server marks it as an Non-active client). These messages can be send by the script and states can be determined by the script itself.

**Q.) How is data stored in the TAM (includes points such as data cant be stored in plain text; mapping of stored data, result)**

**Approach**: Data is always having a one safe copy on Server if in case a task need a part of data, it is copied on to the client's shared memory space and a map of data[5] is stored onto server.

   Similarly map of task[6] is stored on server to point the work of each of
Q.) Need to make sure not using systems resources to an extent that system is becoming slow or is getting out of resources. (Processor usage judgment)

Approach: When ever a client is running a task script at regular interval checks for the logical / physical memory usage (real / private memory), cpu threads, %cpu usage, processes, data read, data write, network usage and pre-define threshold should be set which when exceeded is informed to the server and steps like halting the process and shifting it on some other client with more resources can be taken.

Also the system whose resources are put to use is always given the option to stop the shared tasks running on it. A copy of the current state of computation etc. is sent to the server and the resources are released.

Thus regular checks if detect running out of memory or resources for the task steps like providing virtual resources or shifting the further task on a more resourceful client can be taken.

Q.) What if System Crash Down (What if a system connected crashes down which included shutting down of a system forcefully, manually, getting disconnected by any means from the network.)

Approach:
1. If shutting down (manually): A copy of the current state of computation, results (if any), data (if any on clients shared memory) etc. is sent to the server and the resources are released.

2. If shut down forcefully or in other words system got disconnected from the network anyhow: In this case since we are having a map of task and map of data on Server is updated and the task is send to a new client.

Q.) What happen when client want to disconnect from the AppRes?

Approach: Copy of data is taken/shifted onto another Client and corresponding entries related to this client is updated in all the tables.
**Terminology**

[1] **AppRes Community**: It's a cluster of clients connected to the Server making a small community of resources sharing systems.

[2] **Task Table**: It's a table that contains the id's of clients which are doing a particular job. While a client is doing a job it gives a regular acknowledgment to Server to make sure that client is in Active state and has not gone into Offline or Non-Active state due to any reason. After completion of a task by client the result (if any) is brought back to server where further work is taken care of.

NOTE: Every time clients acknowledge the server it also sends info about TFM and system state (are too much of resources been used by task and making over all system too heavy ( ??? in computer language what will be ): if so then task should be halted till then result should be backed in server and the remaining task should be continued on a client which have more resources and can handle.

On completion of task from different clients task-table mapping that client is updated to task finished and client is marked Inactive / Non-active / Offline depending on the state.

Once all the clients corresponding to a particular task finishes their work the entry from task table is moved to Completed Task Table.

[3] **Completed Task Table**: It's keeps the entry of the task which got completed (can be taken as log) contains
- Time when task starts
- Time when it finishes
- Result if any (and where is it stored / kept)
- Total clients used
- Number and id of Clients failed (which will help to improve the system)
- Time take by each client (so to judge the efficiency of a client system which will intern can be used by AppRes developers to make the task assignment better)

[4] **Total Available Memory (TAM)**: is the memory available to the server from n clients which are connected and sharing and average of (say) m MB i.e TAM = n*m MB

[5] **Map of Data / Data Map**: It's a table present on Server side that keeps track of which part of data is uploaded / present on which client id, thus helping to track the shared data.

[6] **Map of Task / Task Map**: It's a table for each task present on Server
side that keeps record of all those clients which are performing this task.

[7] **Total Free Memory** (TFM) = 1024 MB (or X MB) + memory given by client to AppRes for using

Reasons for having this definition: shortage of memory might cause the client to ask AppRes to temporarily stop sharing the resources, as a result if we well in advance stop the clients resources utilization we can prevent the sudden termination of task (if any, were supposedly running on client machine)