Google Summer of Code 2009 Summary: re-ordering server lists.

Author: Jacob Thebault-Spieker
Mentors: Jeffrey Altman, Derrick Brashear
Presented by: Jeffrey Altman

Current ranking solution

- Server rank is currently done by numerical proximity of the ip address of a server to the ip address of the client.
    - Same machine
    - Same sub-net
    - Same class network
    - Everyone else
- This scheme is based on outdated network architecture assumptions, which no longer apply to a globalized internet.
- This server ranking scheme is ineffective at assigning an appropriate rank, because more often than not, these architecture assumptions do not hold.

Goals of the project

Devise a new ranking method by:

- using rx peer data that is already being collected to rank servers based on packet round-trip time
- taking into account "throughput" data derived from the RXAFS_StoreData and the RXAFS_FetchData RPC calls.
- using both of these locally collected statistics to order the local server lists that the cache manager maintains
- re-ranking the server lists every so often in order to take into account new data.
- implementing this on both the unix and Windows platforms.

Specifics - Rx data collection

- Because rx is a peer-to-peer protocol, both clients and servers already collect rx peer data, and no distinction is made at that level.
- Currently, the rx data is only exposed the tool 'rxdebug' and is not available to either the server or the client.
- The function rx_GetLocalPeers() exposes the data that the client currently collects to allow the cache manager to use the round-trip time (rtt) as a metric to rank servers.
- The round-trip time measurement accounts for RPC processing time when it's being measured.

Specifics - RPC throughput data

- The RPC throughput data is collected while processing two RPCs:
  - RXAFS_StoreData
  - RXAFS_FetchData
- These RPCs send data across the network, one originating from the client, and one originating from the server.
- The amount of data sent or received, and the time each operation took, is collected.
- Data sent(or received) / the time the whole operation took = throughput
- the throughput is then accounted for in the server rank.

Specifics - RTT Implementation

- The ranking technically allows for a rank between 1 and 65,535, but currently the worst rank is ~50,000.
- Instead, the new ranking scheme (when there is only rtt data) ranks along an ln curve based on the equation:

  rank = (int)(7200 * ln(rtt) / 5000) * 5000 + 5000

  - dividing by 5000 and then rounding to the nearest integer, then multiplying by 5000 essentially rounds to the nearest multiple of 5000.
- gives a rank between 5,000 and 65,000, where each increase of 5,000 in the rank is twice that of the previous 1000
  - (1 to 5,000, 2 to 10,000, 4 to 15,000, 8 to 20,000, etc)

Specifics - RTT Implementation cont'd.

- Currently, administrators have the option of overriding the rank by setting their own.
- If the administrator rank is set, said rank will be adjusted by up to 5,000. The following formula is used:

  rank = (int)(623 * ln(rtt) / 10) * 10 + 5

  - dividing by 10 and then rounding to the nearest integer, then multiplying by 10 essentially rounds to the nearest multiple of 10.
- modifies rank up to a rank 5,000 away from that which is set by the administrator. Everytime rtt doubles, the rank changes by 431.8 (the extra 1.8 is rounded away)
  - (1 to 5, 2 to 435, 4 to 870, 8 to 1,300, etc)

Specifics - Throughput Implementation

- When taking into account throughput, currently RTT and throughput are considered as important as the other, so the average of the two ranks is taken, and this becomes the new rank assigned. The throughput rank is calculated in two parts:

  scale = (int) -e^{60,000} / (lowestThroughput - highestThroughput)

  and

  rank = (int)(scale * log(throughputl) / 5000) * 5000 + 5000

- The scale is calculated based on the highest and lowest known throughput for the current cell. This is then used to setup a logarithmic curve similar to that of the RTT curve, but limited by the known throughuts for the specific cell.

Accomplishments

- The RTT implementation is currently in Gerrit, and in testing on the Windows platform
    - Not yet implemented for unix, coming soon.
- The Throughput-aware implementation for Windows is in Gerrit, but needs to be modified before it is safe to test.
    - Not yet implemented for unix, coming soon.
- Checks every 10 minutes to see if new data has been collected, and if so, re-ranks the server in question.