Performance Evaluation of Game Server Protocols

Umang Desai

CS 675

Spring 18

Summary: This report is an analysis of two protocols commonly used in game servers, RMI and Sockets. I have made two separate java programs with the same functionality one using RMI and other using Sockets.

My analysis is done based on calling the same functionality and measuring the time taken to execute the same using RMI and Sockets. I then compare the both based on the findings. The time measured is in nanoseconds.

Analysis:
We will consider each functionality(move, location, status) separately and collect a few performance samples for each protocol.

MOVE

Command	Socket	RMI
`move up 5`	571479	1326548
`move down 4`	363909	1302352
`move left 3`	350231	752486
`move right 5`	386810	800837
`move down 8`	365340	795608
`move up 2`	567860	885206

LOCATION

Command	Socket	RMI
`location`	353997	1606486
`location`	272910	759652
`location`	284311	873600
`location`	267492	689933
`location`	300628	685194
`location`	311343	723375

STATUS

Command	Socket	RMI
`elements`	569645	1453816
`elements`	341308	616216
`elements`	302776	758453
`elements`	309772	747509
`elements`	298522	688178
`elements`	257680	712835

Here we can see that SOCKET is relatively faster than RMI. A peculiar pattern I spotted was that every time we use sockets or RMI (and more commonly with RMI), after a gap or for the first time the time take is much higher than the calls which follow.

One reason why I think socket is faster than RMI is because of the overhead calls it has to make in order to marshal and un-marshal the data. Also it is obvious that RMI is a better option for systems which have a object oriented approach. If a system is message passing based or raw data based, then we should probably choose sockets.