May 2022

Responsive User Interface Updates for Remote Computation

Xu Cao

Responsive User Interface Updates for Remote Computation

ABSTRACT

Remote display protocols send local user inputs such as keystrokes and mouse clicks received at a client device to a remote server and update the local display based on responses received from the remote server. This can sometimes result in unacceptable latency between user commands and display actions, e.g., when performing computation heavy operations such as computer-aided design, version control for large software repositories, etc. This disclosure describes techniques to generate a low-latency response to user inputs by providing a display module and a compute module within client software that executes locally. The display module locally processes display-related user commands to deliver a low latency response to user inputs. The compute module transmits compute-related commands to a remote server for remote computation, the results of which are displayed once they become available.

KEYWORDS

- Remote display protocol
- X11 forwarding
- Software latency
- Display lag
- Display latency
- Remote computation

BACKGROUND

A thin client coupled to a server is a common computing paradigm where a remote, high capacity server performs heavy duty computations for a local client with relatively low computing capacity. The local client is suitable for simple tasks such as viewing the results of the computation done by the remote server, receiving and relaying user inputs via an input device, etc.

Bandwidth constraints and security concerns often make it infeasible to move large amounts of data (e.g., a code repository) between a client and a server. Generic remote display protocols such as X11 send local user inputs such as keystrokes and mouse clicks to the remote server and update the display based on results received from the remote server. This can sometimes result in unacceptable latency between user commands and display actions.

While latency impairs user workflow for any software that forwards local user inputs to a remote server, it particularly affects complex software such as computer-aided design (CAD) tools where an engineer might have to wait several seconds to move an object across the screen; version management tools where a developer might have wait several seconds to difference two or more commits; etc.

DESCRIPTION

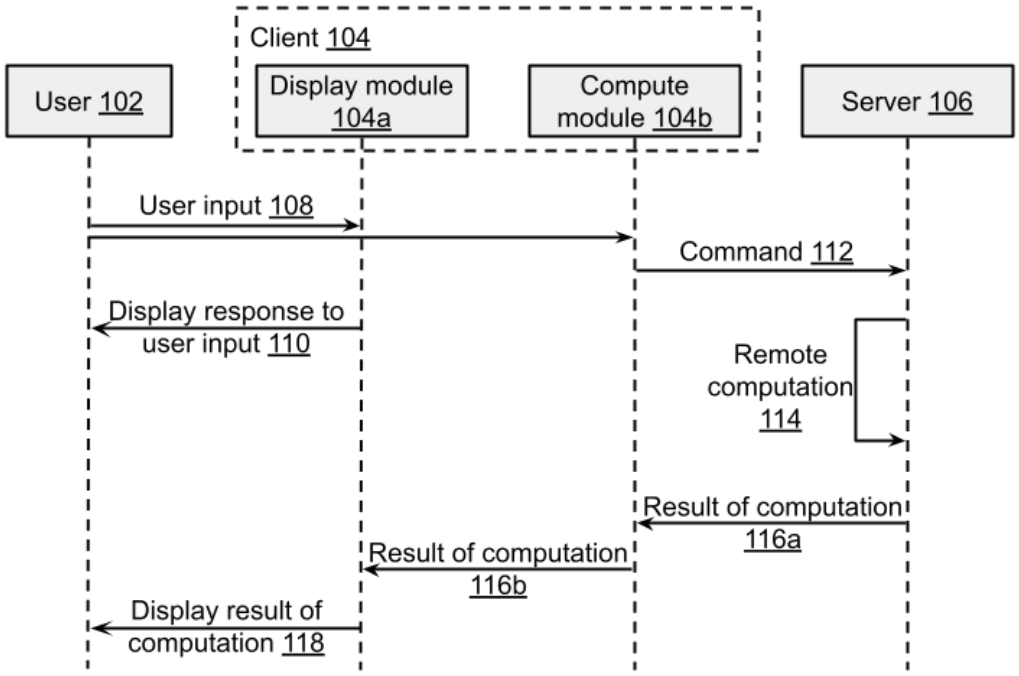

Fig. 1: Responsive user interface updates for remote computation

This disclosure describes techniques to provide a quick response to certain user inputs at a client device on the display of the local screen. As illustrated in Fig. 1, client software (104) includes two modules, one for display (104a) and one for computation (104b). Input (108) from the user (102) is handled by each module in a manner such that display-related user commands, e.g., keystrokes, mouse actions, etc., are processed by the display module while compute-related user commands are processed by the compute module.

Display-related commands, e.g., keystrokes, mouse actions, touchscreen inputs, etc., are processed locally by the display module to generate a rapid, low latency response (110) to the user input. Compute-related user commands (112) are communicated to a server (106), which performs a remote, possibly time-intensive computation (114). The results of the remote computation (116a) are returned to the compute module, which in turn forwards the results to the display module (116b), which displays the results of the remote computation (118).

Separation of display and computation into distinct modules can be standardized, e.g., made plug-and-play, such that the modules can fit, or be imported into, different software. Software developers can thereby provide software with near-zero display latency without additional effort, by simply importing standardized components. Users benefit from a rapid response to user commands.

CONCLUSION

This disclosure describes techniques to generate a low-latency response to user inputs by providing a display module and a compute module within client software that executes locally. The display module locally processes display-related user commands to deliver a low latency response to user inputs. The compute module transmits compute-related commands to a remote server for remote computation, the results of which are displayed once they become available.