Heather Linich

Dec 1, 2014

Arrakis: The Operating System is the Control Plane

OSDI 2014

Arrakis is a newly developed operating system that divides the typical role of the operating system into two parts. Within Arrakis, increasing amounts of I/O operations bypass the kernel due to the program having direct contact with virtualized I/O devices and the kernel itself is reconfigured in order to afford disk and network safeguards without the kernel having to facilitate each operation.

There are many factors that must be considered when the kernel is eliminated from the data path of I/O operations. Among them, programs must still be afforded the same levels of security as traditional operating systems.

Arrakis removes some types of overhead completely while lessening the effect of other types due to leveraging hardware assistance in order to eliminate kernel intervention from the data plane. Kernel crossing and scheduling overhead is completely eradicated as packets are now distributed straight to user space. Processing of the network stack is also significantly simplified since it is not required anymore to demultiplex packets for separate programs. The user level network stack also does not have to substantiate user parameters as thoroughly as a kernel implementation.

Many traditional operating systems spend significant resources processing the network stack. In Arrakis’ network stack, copying data to and from user stipulated buffers is where most of the processing cost arises. Upon receipt, new data is put into a network buffer by the network hardware. It is then copied into the location identified by the POSIX (Portable Operating System Interface. POSIX are criteria for retaining compatibility among operating systems – *Wikipedia*) read call. Data is transferred into a buffer that can then be put into the network hardware queue. After completion, the POSIX write call is able to return which permits the user memory to be reused before any new data is sent. Arrakis has a native interface that is able to maintain authentic zero copy I/O.

There are three main design objectives when creating Arrakis. The first is to lessen kernel participation for data plane processes. Arrakis is intended to significantly limit or eliminate kernel intervention for the majority of I/O operations. I/O calls are directed to and from the program’s address space with no kernel participation but also without forfeiting any security or isolation features. Arrakis also provides transparency to the programmer. It can considerably increase performance without necessitating changes to applications that are written to the POSIX API. Finally, Arrakis offer suitable OS and hardware abstractions. The abstractions should be adaptable enough to proficiently sustain a broad range of I/O patterns, scale well for multicore systems and maintain program requirements for load balance and locality.

A crucial element in the Arrakis development is a hardware independent layer that supports virtualized I/O. An interface has been developed between the program and the control plane and its purpose is to ask for resources as needed from the system and to guide I/O flows between user programs. Applications within Arrakis transfer network packets by communicating with the hardware. The data plane interface is executed in an application library permitting it to be designed alongside of the application itself.

The library in Arrakis offers two different interfaces to programs: a native interface that marginally deviates from the POSIX specifications in order to support true zero copy I/O and also a POSIX compatibility interface to sustain unmodified programs.

Hardware does not have to accomplish protocol processing, it only requires multiplexing. A user space network stack exists above the data plane interface in order to exhaust latency and throughput. There are three separate features of packet communication and reception. Packets are transmitted asynchronously between main memory and the network by traditional DMA methods which uses rings of packet buffer descriptors. The program moves possession of a transmit packet to network hardware by enqueueing chained buffers onto the hardware descriptor rings. A packet would be received by reversing this procedure. Arrakis manages asynchronous notices of events by using doorbells linked to queues. When programs are currently running, doorbells are transported from hardware to user programs through hardware virtualized interrupts. If the program is not running, doorbells come through the control plane to summon the scheduler.

The Arrakis team evaluated their operating system against Linux in a number of ways. Among them are that Linux separates the network stack and application but this offers incomplete information about packet processing. It also makes it challenging to allocate threads to the correct CPU core. This produces cache misses and socket lock contention which account for a great deal of Linux’s overhead. However, in Arrakis, the program controls the entire packet processing procedure. This in turn means that the network stack does not have to obtain locks and packet data and information is permanently accessible in the correct processor cache.

Arrakis can be a useful tool for those looking for superior operating system performance. Due to the division of procedures and the fact that Arrakis only keeps the kernel minimally involved, users would have an efficient and effective operating system to use.

Since there is such a clear separation of processes and procedures within the operating system, Arrakis could also offer the user more reliability and security. Arrakis has also shown that their model can be less resource expensive through their thoughtful design, such as reducing the processing of the network stack. This could possibly allow it to run on systems that have fewer resources to spare on operating system tasks.

The Arrakis team has already shown that their product could be a respectable rival for Linux and if continually developed, Arrakis could easily catch on to more and more home and commercial consumers. Arrakis takes operating system innovation to a new level and could help pave the way for even more forward thinking and efficient operating systems in the future.