DISS. ETH No. 23235

End-to-End Considerations in Unification of High-Performance IO

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by
Animesh Trivedi
Ing. dipl., ETH Zurich
born on August 02, 1986
citizen of the Republic of India

accepted on the recommendation of
Prof. Dr. Thomas R. Gross, examiner
Dr. Bernard Metzler, co-examiner
Prof. Dr. Torsten Hoefler, co-examiner
Prof. Dr. Edouard Bugnion, co-examiner

2016

Abstract

The performance of modern distributed storage and computing frameworks considerably depends on the IO performance of the many storage and network devices involved. Fortunately, these IO devices have undergone a rapid transformation in the past decade and are now capable of delivering multi-Gigabits/sec bandwidths and ultra-low IO latencies. However, in contrast to IO devices, the performance improvements of single CPU have stalled in the same time period. Hence, the traditional notion of a single fast CPU connected to multiple slow devices no longer holds. Yet, IO stacks are still designed to optimize the CPU time by executing multiple services and routines on a *fast* CPU while a *slow* IO operation is in progress. This situation has led to a *CPU-IO* performance gap, where the CPU's inability to keep up with the execution of thick software stacks and OS routines during a fast IO operation on high-performance network and storage devices limits the performance delivered to data-crunching applications. Multiple research efforts from industry as well as academia have been launched to improve this situation by reducing the hardware and software overheads by providing better IO interfaces, efficiently managing IO resources, and leveraging manycore CPUs for IO processing. However, these efforts exclusively either target the network or the storage stack but not the combination of both.

In this thesis, we address this performance gap and advocate to take a holistic approach towards managing resources, data flows, and devices (network or storage) to form end-to-end data flows in a distributed setting. We first quantify the software and OS overhead in IO operations and and argue to reduce it by building upon the high-performance networking principle. The general philosophy of the principle is to recognize and separate the slow control setup from the fast data access path, and involve CPU/OS in the former only selectively in managerial roles. In the thesis framework, we extend the separation philosophy from networks to storage devices by identifying common themes in the evolution of their software stacks. After identifying common high-performance IO properties, we make a case to unify the network and storage stacks. We then design and build a proof of concept FlashNet, a unified software IO stack that uses high-performance networking abstractions and semantics to access remote storage. In accordance with the original separation philosophy, FlashNet allows the allocation and translation of both

network *and* storage resources prior to a remote storage access.

We then expand the philosophy and demonstrate how to separate the resource setup from fast data accesses in a distributed environment where resources might be spread across multiple machines. We introduce RStore, a distributed in-memory data store that achieves distributed separation using its unique memory-like API and storage abstractions. We quantify the effectiveness of this approach by developing two distributed data-processing applications on top of RStore, namely a distributed key-value sorter (RSort) and a distributed graph-processing engine (Carafe). They both perform well; RSort, for example, outperforms Hadoop TeraSort by a margin of $8-10\times$ on our 12-machine cluster. At the end of the thesis, we document our experience and give recommendations for system builders on how to leverage the separation principle to cut through the thick IO abstractions and layers to design and implement high-performance distributed data processing applications.

Zusammenfassung

Die Leistungsfähigkeit moderner verteilter hängt in hohem Masse von der IO-Leistung der eingesetzten Speichergeräte und des Netzwerkequipments ab. Diese IO-Geräte haben im letzten Jahrzehnt eine schnelle Weiterentwicklung erfahren; sie bieten heute eine Zugriffsbandbreite von mehreren Gigabits pro Sekunde bei sehr geringer Latenz. Im Gegensatz zu den IO-Geräten hat sich die Leistungsfähigkeit einer CPU im genannten Zeitraum kaum erhöht. Damit ist die bisherige Annahme einer Rechnerarchitektur mit einer einzelnen schnellen CPU, die mit mehreren langsamen IO-Geräten interagiert, nicht mehr gültig. Allerdings sind IO-Stacks immer noch unter der Annahme gestaltet, die CPU-Zeit optimal auszunutzen, indem eine schnelle CPU mehrere Dienste und Routinen abarbeitet, während eine langsame IO-Operation zeitgleich abläuft. Diese Situation hat zu einer Disparität zwischen CPU- und IO-Performance geführt, in der die CPU nicht in der Lage ist, komplexe Softwarehierarchien und Betriebssystem-Routinen während einer schnellen Netzwerk- oder Speicher-IO-Operation auszuführen. Dies limitiert die für Applikationen zur Verfügung stehende Leistungsfähigkeit. Es wurden eine Reihe von Forschungsprojekten in industriellem und universitärem Umfeld gestartet, die diese Situation verbessern wollen, indem die Hardware- und Software-Overheads durch bessere IO-Interfaces, effizientes Management der IO-Ressourcen und die Nutzung von Vielkern-Prozessoren reduziert werden sollen. Diese Aktivitäten beschränken sich jedoch auf entweder den Netzwerk- oder den Speicherzugriff und nicht auf eine Kombination beider Dienste.

Die vorliegende Arbeit adressiert diese Disparität und plädiert für einen holistischen Ansatz, in welchem Ressourcen, Datenströme und Geräte (sowohl Netzwerk- als auch Speichergeräte) so verwaltet werden, dass sie in einem verteilten System zu einem Ende-zu-Ende-Datenfluss formiert werden können. Einleitend wird der Software- und Betriebssystem-Overhead für IO-Operationen quantifiziert und daraus der Vorschlag abgeleitet, diesen mit Designprinzipien von Hochgeschwindigkeitsnetzen zu reduzieren. Das Grundprinzip dieses Ansatzes ist es, zeitunkritische Kontrolloperationen vom zeitkritischen Datenzugriff zu separieren und dabei CPU und Betriebssystem nur bei Kontrolloperationen zu involvieren. Im Rahmen der vorliegenden Arbeit wird dieses ursprünglich für die Einbindung von Netzwerkgeräten eingeführte Separationsprinzip auf Speichergeräte erweitert, indem gemeinsame Problemfelder bei der Evolution

der zugehörigen Software-Stacks identifiziert werden. Aus der Identifikation gemeinsamer IO-Eigenschaften beider Stacks wird der Vorschlag abgeleitet, diese in einer Implementierung zu vereinen. Dieses Konzept wird anhand der FlashNet-Implementierung, die Abstraktionen und Semantiken von Hochgeschwindigkeitsnetzen für den Zugriff auf entfernten Massenspeicher nutzt, exemplarisch umgesetzt. FlashNet setzt die Philosophie der Trennung von Kontroll- und Datentransferoperationen um und erlaubt damit eine Reservierung und Bereitstellung von sowohl Netzwerk- als auch entfernten Massenspeicherressourcen vor dem effizienten Zugriff.

Im Weiteren wird die Designphilosophie der Trennung von Daten- und Kontrolloperationen verallgemeinert und auf ein verteiltes System angewandt, in dem die Ressourcen über mehrere Endpunkte verteilt sind. Mit RStore wird ein verteilter, hauptspeicher-residenter Datenspeicher eingeführt, der dieses Prinzip durch den Einsatz eines neuen Applikationsprogramm-Interfaces und von Speicherabstraktionen realisiert. Die Effektivität dieses Ansatzes wird anhand von zwei neu entwickelten RStore-Applikationen quantifiziert: RSort, ein verteilter Key-Value Sortieralgorithmus und Carafe, eine verteilte Anwendung zum Prozessieren von Graphen. Beide Anwendungen erreichen ausgezeichnete Performance - zum Beispiel übertrifft RSort das bekannte Hadoop TeraSort um den Faktor acht bis zehn auf dem verwendeten 12-Knoten-Cluster. Die vorliegende Arbeit schliesst mit einem Resümee der gefundenen Erkenntnisse und mit Empfehlungen für Systemarchitekten, wie mittels des Separationsprinzips aufwendige IO-Abstraktionen und Schichten vermieden werden können, um hoch leistungsfähige verteilte Applikationen zur Datenverarbeitung umsetzen zu können.