LISTIC

Offre de thèse 2021-2024

Titre	« Nouveaux mécanismes systèmes et réseau pour la prise en charge des flux de données massif en apprentissage » "New network systems mechanisms to support Massive Machine Learning
Niveau du stage	Master 2ème année / Ingénieur 5ème année
Date de début/ fin	Octobre 2021 (3 ans)
Ville, Pays	Annecy-le-Vieux, *France*
Laboratoire	LISTIC - Laboratoire d'Informatique, Systèmes, Traitement de l'Information et de la Connaissance - https://www.univ-smb.fr/LISTIC
Description du sujet de stage	Cette thèse vise à développer de nouvelles primitives système / réseau pour améliorer l'utilisation du réseau et des ressources informatiques utilisées dans les pipelines de traitement de données d'apprentissage automatique. Nous étudierons comment adapter le système et le réseau pour le traitement des flux de données. Les applications pilotées par Machine Learning ont des caractéristiques particulières qui peuvent entrer en conflit avec la prise en charge système de base fournie par les systèmes d'exploitation, la virtualisation ou le réseau. La thèse portera sur des approches innovantes en matière de placement de données, en particulier sur les systèmes de caches et de dépôt labels/contenus distribués. Vers cet objectif, le paradigme réseau orienté contenu (ICN) et le routage par nom (NDN) sont des approches très prometteuses. Le concept fondamental ici consiste à pousser certaines des fonctionnalités des systèmes très populaires de cache applicatifs comme REDIS, ou MemCached, dans la fonctionnalité de cœur du réseau, et ainsi de développer une nouvelle sémantique de synchronisation de cache distribuée basée sur cette architecture réseau. Ceci permettra de voir le déploiement d'un pipeline de traitement non pas comme un placement sur des machines identifiés de composants de traitements et la mise en relation de ceux-ci par le biais d'un routage statique, mais comme la définition d'un espace adéquat de nommage des contenus échangés, que les composants de traitement utiliseraient avec une approche publieur/souscripteur (publish/subscribe). This Ph.D. thesis aims to develop new system / network primitives to improve the utilization of network and computing resources used in Machine Learning data processing pipelines. We will study how to adapt the system and the network for the processing of data flows. Machine Learning driven applications have special characteristics that can conflict with the basic system support provided by operating systems, virtualization or the network. The thesis will focus on innovative approaches in data placement, on caching systems and distributed content repositories. To this end, the Information Centric Networking (ICN) paradigm and name-based routing (NDN) are very promising approaches. The fundamental concept here is to push some of the functionality of very popular application caching systems like REDIS, or MemCached, into the network core functionality, and thus to develop a new semantics of distributed cache synchronization based on this network architecture. This will make it possible to address the deployment of processing pipelines not as the placement on identified machines of processing components and the linking of these through static routing, but as the definition of an adequate naming space for exchanged content, which processing components would use with a publish / subscribe approach.
Sujet détaillé	Driven by advances in computation infrastructure processing large amounts of data has become a key scientific tool across a large spectrum of scientific domains, ranging from physics, chemistry, geosciences, biology and even humanities. Big data processing needs have motivated the integration into the scientific infrastructure of data centers and high-performance computing clusters. Such computing architectures are accessible concurrently by multiple users and implement large scale distributed and parallel processing systems that can integrates millions of threads cooperating together through a networked infrastructure. Managing this distributed architecture leads to very complex challenges across distribution of the processing, data placement, resilience to failures, and more [GHMP08]. The described issues become even more complex due to the inherent trade-offs between objectives and constraints of modern applications. On one hand, the infrastructure has to guarantee that the processing is done as fast as possible to cope with ever-growing incoming data rates. On the other hand, technical constraints like availability of specific types of hardware (e.g., CPUs or GPUs), available RAM memory or disk space, transfer delay from a data producer to a consumer, and so on, have to be combined with financial constraint (i.e.,

price of resource), environ- mental impact (i.e., reduce the energy footprint of processing), and increasing privacy concerns (e.g., privacy-aware and federated learning). Finally, data processing pipelines generates strong fluctuations in the quantity of needed resources, which combines to failures in the processing or storage that, unfortunately, are common on such heavily loaded infrastructures.

To enable machine learning developers to exploit large scale computing clusters minimizing their need to cope with the presented issues requires developing advanced application management systems. Traditionally, developers of machine learning applications had to deal with a single process and was interacting with the underlying computing hardware through the Hardware Abstraction Layer (HAL) virtualization provided by the operating system. To cope with the new data processing needs along with the distributed nature of the hardware requires the development of consistent Network Operating Systems (NOS) that shield application developers from the details of the hardware and the system implementations.

Many Scientific Workflow Management Systems (SWMS) have been developed to simplify the task of deploy such processing pipelines. For example, Hadoop [Bor07] is a popular SWMS that provides through a java language interface the primitives for deploying a Map-Reduce type of processing over a computing cluster. Similarly, pytorch [PGM+19] and tensorflow [ABC+16] are python-based tools that provide support for deep learning neural network processes at scale. All these systems are based on specific HALs that enables a centralized scheduler to map processing task to hardware resources.

Nevertheless, a fundamental issue that is front of all of these SWMS is the need for a shared memory space among threads to enable different workers to flexibly (and rapidly) exchange information. This memory space should ensure address/naming consistency across distributed threads that might be running over distant computers, so that a worker needing some information for its execution can retrieve it independently of the physical location of the accessed data. Moreover, this memory space should be efficient — i.e., accessing data should be fast while not wasting memory and network bandwidth. In a nutshell, the data should be stored close to the workers that use them, and the memory content should follow the dynamics of data usage patterns during the calculation. These requirements are very similar to what consistently cached Non-Uniform Memory Access systems (ccNUMA) tries to achieve. However, we need to expand this to memory across remote computers, accounting for the dynamicity introduced by failures as well as ensuring a consistent and meaningful addressing/naming space. This thesis project targets the development of such system.

Ph.D. thesis work. In the past years, publish/subscribe (pub/sub) systems (e.g., [The14], RabbitMQ [rab20], or [Lig17]) have attracted interest as a mean to implement distributed information distribution with consistent addressing/naming. These systems make possible for an information source to publish generated data over a distribution channel identified by a label, and for information sinks/consumers to subscribe on labelled channels and receive consistent updates. This means that pub/sub systems can potentially address the need for a distributed ccNUMA with a consistent naming space defined as channel topics, becoming the needed distributed middleware for SWMS memory exchanges. While the emphasis in pub/sub is on the diffusion of information over channels it becomes apparent that the fundamental underlying problem is a question of cache synchronization and of data consistency: Data published over a channel is stored in a cache in the pub/sub broker and registration to a channel is equivalent to a cache sync request. The pub/sub mechanism is there to guarantee the consistency of the cache sync through forwarding in an atomic way such that any brokers' cache update propagates to all subscribed caches.

Two issues arise: First, each pub/sub broker is a single performance bottleneck and point of failure. When the number of subscribers and/or the amount of exchanged data increases, the load on the pub/sub broker increases and considerably slows down the synchronization process. Second, pub/sub semantics assume explicit subscriptions. These mechanisms can lead to sub-optimality, whereas it might be preferred to leverage the data temporal and spatial locality in order to do opportunistic preemptive cache synchronization (similar to what is implement in CPU memory caches). One solution to both of these issues is to consider the problem as a distributed cache management system.

Distributed cache management systems also become very popular thanks to implementations such as REDIS [red20] and CacheMem. These systems use cache store < key, $value$ > tuples to retrieve the value stored along with it. These caches could guarantee fast access to stored data with delay less than 1ms. By combining pub/sub systems with distributed key-value caches, one can provide the shared among all threads memory space

that will enable different workers to exchange information. Each key would have assigned to it a publishing channel, and whenever a worker needs a data it asks to its local cache if it has a value stored that is matching the given key. If not, the local cache subscribes to the channel and gets a copy of the value coming from the broker. The key-value caches are already instrumented with memory management systems like Least Recently Used (LRU) replacement policies. Moreover, they enable the use of key access pattern in order to develop opportunistic channel/key subscription that benefit from locality.

However, one core issue remains unsolved. The pub/sub broker can still become the performance bottleneck of the system. In the thesis we aim to leverage the Name Data Networking (NDN) [JST+09, ZAB+14] paradigm in order to overcome it. NDN is an Information Centric Networking (ICN) approach that enables communications among nodes based on data names, decoupling them from their locations. NDN enables access to named content without the need to go through a centralized broker. This is done through the implementation a new routing approach that use name-based routing tables rather than traditional IP addresses based one. Therefore, the combination of key- value caches and the NDN routing protocol can provide a flexible, efficient, and resilient middleware that can be used in machine learning and, more broadly, big data applications to simplify information exchanging in large-scale distributed processing systems. This provides a key element of the Network Operating System we introduced earlier — i.e., the memory access API. We aim to build such a middleware in this Ph.D. thesis.

Further, we aim to build on top of this general framework an efficient implementation of distributed caches. Data replication has been used to reduce the access delay by storing the data close to its consumers. But replication is expensive across three dimensions: a) the amount of storage used, b) the amount of backhaul capacity required to transfer the date, and c) the update costs. Minimizing the cost of contents' diffusion involves combining storage management, routing, and content discovery subject to dynamic changes resulting from popularity variation and network changes. Existing literature minimizes the cost of each one of the above terms separately to avoid coping with the inherent trade-offs between storage efficiency, network efficiency, robustness toward network change and information updates. However, new applications have stringent delay and throughput constraints that needs dynamic tuning of these tradeoffs in order to achieve best content diffusion performance.

Network Coding (NC) techniques have been applied successfully to tackle these problems in different research domains. Our approach in this thesis will be to develop a common conceptual framework integrating the storage, the transmission and the dynamic update of the content. We wish to leverage this framework in order to dynamically adjust the different trade-offs to changes in content popularity, network bandwidth and storage capacities in high performance Named Data Networks. We will achieve this by integrating into the analysis the content discovery phase. Linear Network Coding (LNC) has been independently applied both to cache management [DGW+10] and to network transmission. The main idea of NC is to store or transmit linear combination of contents rather than the content itself. Entangling the transmission and the storage of several contents enables cooperation between distributed entities by combining several linear combinations coming from different sources (caches). LNC can therefore naturally exploit multipath making it very attractive to alleviate the load on the backhaul links. Moreover, network coding has also been applied to provide failure resilience in distributed storage systems, similar to the application of loss protecting codes in RAID (Redundant Array of Independent Disks) systems.

References.

ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pages
265–283, 2016.

[Bor07] Dhruba Borthakur. The hadoop distributed file system: Architecture and design. Hadoop Project Website, 11(2007):21, 2007.

[DGW+10] Alexandros G Dimakis, P Brighten Godfrey, Yunnan Wu, Martin J Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. IEEE transactions on information theory, 56(9):4539–4551, 2010.

[GHMP08] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks, 2008.

	[JST+09] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In Proceedings of the 5th international conference on Emerging networking experiments and technologies, pages 1–12, 2009. [Lig17] Roger A Light. Mosquitto: server and client implementation of the mqtt protocol. Journal of Open Source Software, 2(13):265, 2017. [PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high- performance deep learning library. arXiv preprint arXiv:1912.01703, 2019. [rab20] Rabbitmq. https://www.rabbitmq.com/, 2020. [red20] Redis. https://redis.io/, 2020. [The14] Khin Me Me Thein. Apache kafka: Next generation distributed messaging system. International Journal of Scientific Engineering and Technology Research, 3(47):9478–9483, 2014. [ZAB+14] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, KC Claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. ACM SIGCOMM Computer Communication Review, 44(3):66–73, 2014.
Compétences requises	The Ph.D. candidate should have a master's degree in a Computer Science with focus in Systems and Networks. Good software system development and some knowledge of machine learning algorithms (in particular deep learning) is desirable.
Tuteurs / Contacts	Pr. Kavé Salamatian – kave.salamatian@univ-smb.fr Francesco Bronzino – fbronzino@univ-smb.fr