Useful papers recap
Notes on papers that might be useful
for the development of this thesis

Marco Micera
Politecnico di Torino
marco.micera@gmail.com

1 In-network processing

1.1 In-network data aggregation

1.1.1 Daiet [7]

Solution for partition/aggregate data center applications (big data analysis such as MapReduce [2], machine learning, graph processing and stream processing).
The network controller pushes a set of rules to network devices in order to (i) establish an aggregation tree (spanning tree) and (ii) perform per-tree aggregation.
Inventors claim to achieve a 86.9%-89.3% traffic reduction.

1.2 In-network data storage

Network devices can be used for data storage and caching, using a distributed key-value map.
The network controller can handle system reconfigurations such as switch failures, additions (e.g., new switch onboarding) and removals (e.g., switch firmware upgrade).

1.2.1 NetChain [4]

The network device in charge of storing the distributed storage/cache is a programmable switch: this brings an obvious limitation in terms of storage size, which makes NetChain [4] an acceptable solution only when a small amount of critical data must be stored in the network data plane (e.g., locks). NetChain [4] also processes queries entirely in the network data plane.

1.2.2 IncBricks [6]

This solution makes use of network accelerators attached to programmable switches whenever complicated operations should be performed on payloads.
Supporting multiple gigabytes of memory, network accelerators overcome the limited storage problem typical of programmable switches, which usually have a memory of tens of megabytes.

2 Resource management

2.1 Apache YARN [9]

Apache YARN [9] uses a single resource manager per cluster: for this reason, this solution may not scale up in large clusters with massive amounts of small applications.

2.2 Omega [8]

Omega [8] is a parallel, lock-free and optimistic cluster scheduler by Google.
There is no central resource allocator: all of the resource-allocation decisions take place in the schedulers. This solution makes use of a data structure (called *cell state*) containing information about all the resource allocation in the cluster. Each cell has a shared copy of this data structure, and each scheduler is given a private, local, frequently-updated copy of cell state that it uses for making scheduling decisions.

According to the optimistic concurrency technique, once a scheduler makes a placement decision, it updates the shared copy of cell state with a transaction. Whether or not the transaction succeeds, the scheduler resyncs its local copy of cell state afterwards and, if necessary, re-runs its scheduling algorithm and tries again.

3 Provider interfaces and guarantee provisioning

3.1 Bazaar [3]

Bazaar [3] is a cloud framework that allows client applications to only specify high-level goals regarding their jobs and applications instead of expressing requirements in terms of needed resources. For instance, a tenant can only specify a maximum completion time of its MapReduce [2] job and let Bazaar [3] determine the best resource combination to satisfy the request. This translation is done by two components: (i) a *performance prediction* component that predicts a set of *resource tuples* (each comprising the number of VMs and the network bandwidth between the VMs) and (ii) a *resource selection* component that selects the best (less costly) resource tuple to be used to complete the task.

3.2 CloudMirror [5]

CloudMirror [5] allows client applications to specify bandwidth and high availability guarantees: this can be done by providing a *Tenant Application Graph* (TAG), a directed graph where each vertex represents an application component and links' weights represent the minimum requested bandwidth. An heuristic VM placement algorithm then tries to solve this NP-hard allocation problem.

3.3 Oktopus [1]

This system maps tenant virtual networks to the physical network while respecting minimum bandwidth constrains. Client applications can request two kinds of network: (i) a *virtual cluster*, consisting in N VMs connected to a virtual switch by a bidirectional link of capacity B (switch bandwidth = $N \cdot B$), resulting in a one-level tree topology; or a (ii) *virtual oversubscribed cluster*, composed of a total number of N VMs in groups of size S, with each group connected connected by a virtual switch of bandwidth $S \cdot B$ and groups connected by a root virtual switch of bandwidth $N \cdot B/O$. The latter abstraction allows provider to fit more tenants on the physical network and limits tenant costs.

The mapping is made possible by a logically centralized network manager that is aware of the network topology, residual bandwidth on links (via SNMP), etc.

References

[1] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 242–253, New York, NY, USA, 2011. ACM.

[2] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, CA, 2004.

[3] V. Jalaparti, H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Bridging the tenant-provider gap in cloud services. In *Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, pages 10:1–10:14, New York, NY, USA, 2012. ACM.

[4] X. Jin, X. Li, H. Zhang, N. Foster, J. Lee, R. Soulé, C. Kim, and I. Stoica. Netchain: Scale-free sub-rtt coordination. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 35–49, Renton, WA, 2018. USENIX Association.

[5] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma. Application-driven bandwidth guarantees in datacenters. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 467–478, New York, NY, USA, 2014. ACM.

[6] M. Liu, L. Luo, J. Nelson, L. Ceze, A. Krishnamurthy, and K. Atreya. Incbricks: Toward in-network computation with an in-network cache. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 795–809, New York, NY, USA, 2017. ACM.

[7] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis. In-network computation is a dumb idea whose time has come. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, pages 150–156, New York, NY, USA, 2017. ACM.

[8] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes. Omega: flexible, scalable schedulers for large compute clusters. In *SIGOPS European Conference on Computer Systems (EuroSys)*, pages 351–364, Prague, Czech Republic, 2013.

[9] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 5:1–5:16, New York, NY, USA, 2013. ACM.