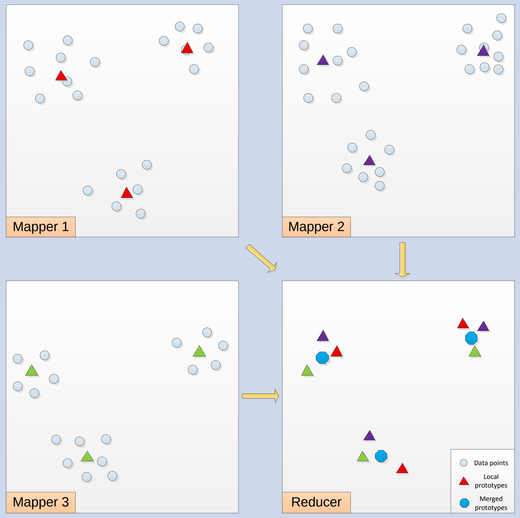
K-Means on Azure

Matthieu Durut Fabrice Rossi Lokad, Telecom ParisTech

Contribution

We present how parallel machine learning algorithms can be implemented on top of Microsoft Windows Azure cloud computing platform. The storage component of the platform is used to provide synchronization and communication between processing units. We report the speedups and scaleups of a parallel K-Means algorithm obtained with up to 200 processing units.

Distributed K-Means

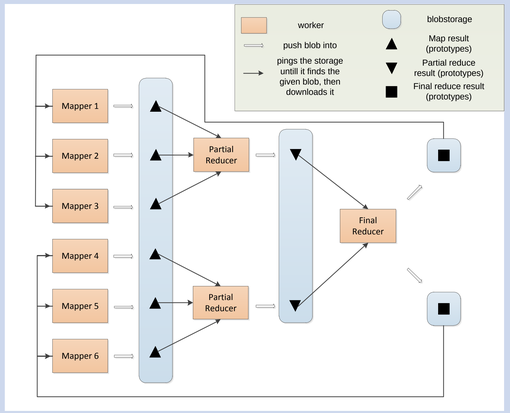


Microsoft Azure

Our prototype is build on top of several components that ease access to storage and workers lifecycle management:

- Azure queues provide a message delivery mechanism through distributed queues. Messages can be returned multiple times, which implies that jobs must be idempotent.
- Azure blobstorage is a key/value pairs storage system. Workers are stateless : all the intermediate and final results are stored in the blobstorage.
- *Lokad-Cloud* framework provides an abstraction layer on storage and workers.

Communication Tree



The reduce phase is a two-step procedure where all the partial reduce jobs are run in parallel.

Cost Modelization

Let :

- N be the number of points in $\mathbb{R}^D$,
- D be the points dimension,
- K be the number of clusters,
- I be the number of iterations,
- F be the time needed to perform one floating point operation,
- B_1 be the time needed to read a double from the blobstorage,
- B_2 be the time needed to write a double into the blobstorage.

In the proposed algorithm, we can roughly model the cost of computation and communication by :

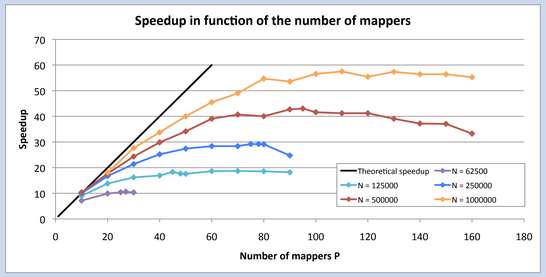
$$T_P^{comp} = \frac{3NKDIF}{P}, T_P^{comm} = 2\sqrt{P}KDI(B_1 + 0.5B_2)$$

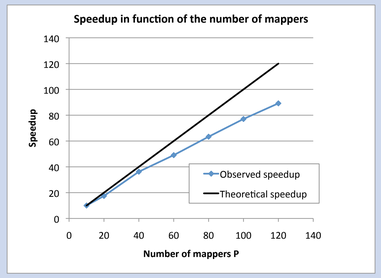
Speedup can therefore be modeled as :

$$S_P = \frac{T_1^{comp}}{T_P^{comp} + T_P^{comm}} = \frac{3NF}{\frac{3NF}{P} + 2\sqrt{P}(B_1 + 0.5B_2)}$$

The optimal number of mapper PU is given by $P^* = (3NF/(B_1 + 0.5B_2))^{2/3}$. For this value of P^*, which depends neither on K nor on D, the total processing time consists in one third of mapping and two third of reducing. Conversely, for N sufficient, communication is negligible and one can achieve good speedup.

Results

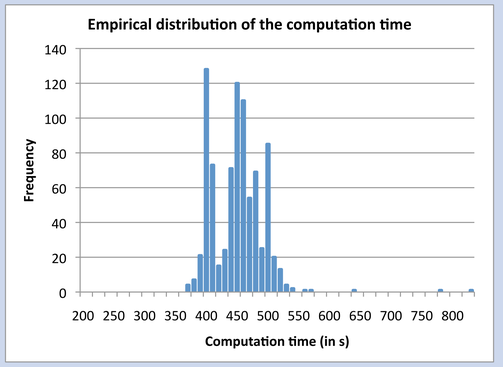


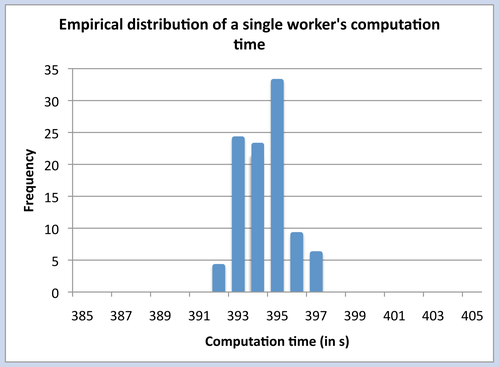


The algorithm has been run for different values of N and P. We set K=1000 and D=1000 in all experiments so latency can be neglected , and experiments were run for 10 iterations to get stable timing estimates.

On the left figure, we fixed N and plot the speedup curve as a function of P, which is relatively flat in a neighboorhood of P^*. But for $P = P^*$, most of the time is spent in communication, resulting in a rather inefficient worker usage.

The algorithm has also been run with much more points per mapper (50,000 points per worker on right figure). Workload in each worker being much more important, communication becomes small compared to computation, and satisfactory speedups and scaleups can be reached, provided no straggler issues are met.





As shown in right figure, each worker performs computation with very stable timing. But some workers may be slowed-down during a short period of time, as shown in worst cases of left figure. This mechanism has little effect on performance while running 100 workers, but may become very critical while running thousands of machines.

Conclusion

Communication costs induce a potential bound on scaling for cloud applications. As anticipated by the model, communication takes more time than computation while running P^* mappers. Conversely, heavy workload on each worker leads to very satisfactory speedups while running up to 150 mappers. Beyond this scale, straggler issues may become a bottleneck, and should be handled.

References

[1] Dhillon, Inderjit S. and Modha, Dharmendra S. A Data-Clustering Algorithm on Distributed Memory Multiprocessors In Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD

Acknowledgement

Special thanks to the Microsoft Bizspark *One* team for their support.