Enhanced approach to compute entity relationships over MapReduce

{ Harsh Wardhan Agarwal , Rishabh Srivastava , Ritu Agarwal }

Stony Brook University

Problem Space

Identifying patterns and discovering hidden relationships with the help of data analysis is a key factor that many technology giants leverage today to their profit and scale up their businesses. Not only in leveraging the business, such analysis is also crucial to many domains such as the health industry where it can be used to find patterns and understand relationships between genes of different species. The theory can potentially be extended to many such domains. But a major challenge our analysis systems face today are not because of unsophisticated algorithms, but their computation limits. The data that needs to be analysed is so huge that it is practically impossible to perform the entire computation without a distributed architecture

Contributions

In our work, we propose an idea that not only leverages the advantages of MapReduce but also parallelises the MapReduce tasks themselves by breaking the bigger goal into smaller sub problems, identifying the tasks that can be run independently and perform multiple and distinct MapReduce tasks parallely to reduce the computation time drastically.

Results

Data	Serial MR (sec)	Parallel MR (sec)
100k (4.1MB)	14.60	11.81
200k (9.9MB)	29.27	26.17
300k (15MB)	38.45	34.96
400k (20.1MB)	48.21	45.20

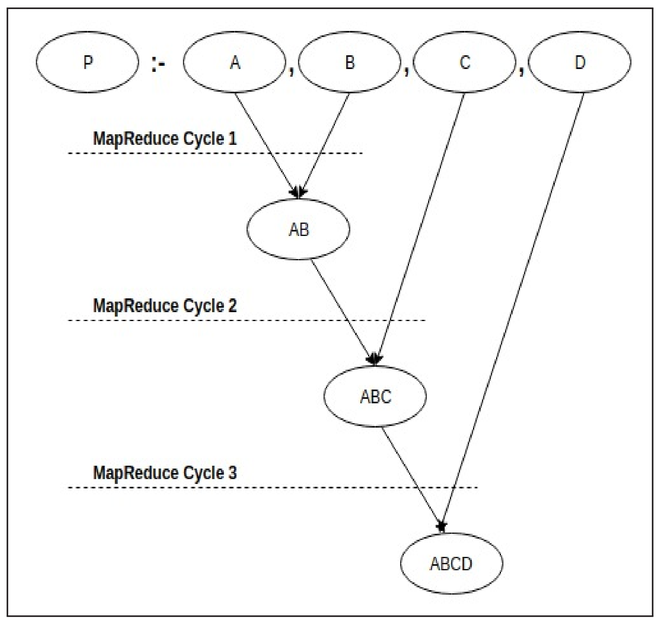
Conclusion

Here, we have just parallelized few map reduce cycles based on the number of independent operations. With these few operations which have been parallelized, we have seen reduction in the time duration in the order of a few seconds. In a practical scenario, where we have zeta bytes of data being processed on a daily basis, we will be able to see drastic reduction in the processing times when operated in a parallel fashion and where we have lots of independent relations which can be parallelized.

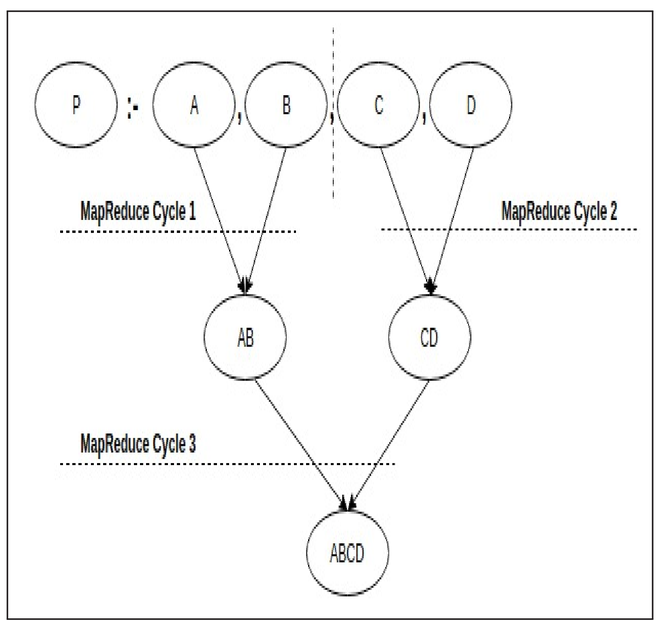
Query

ProductSales(X,Y,Z) :- Buyer(X), Purchased(X,Z), Seller(Y), Sold(Y,Z)

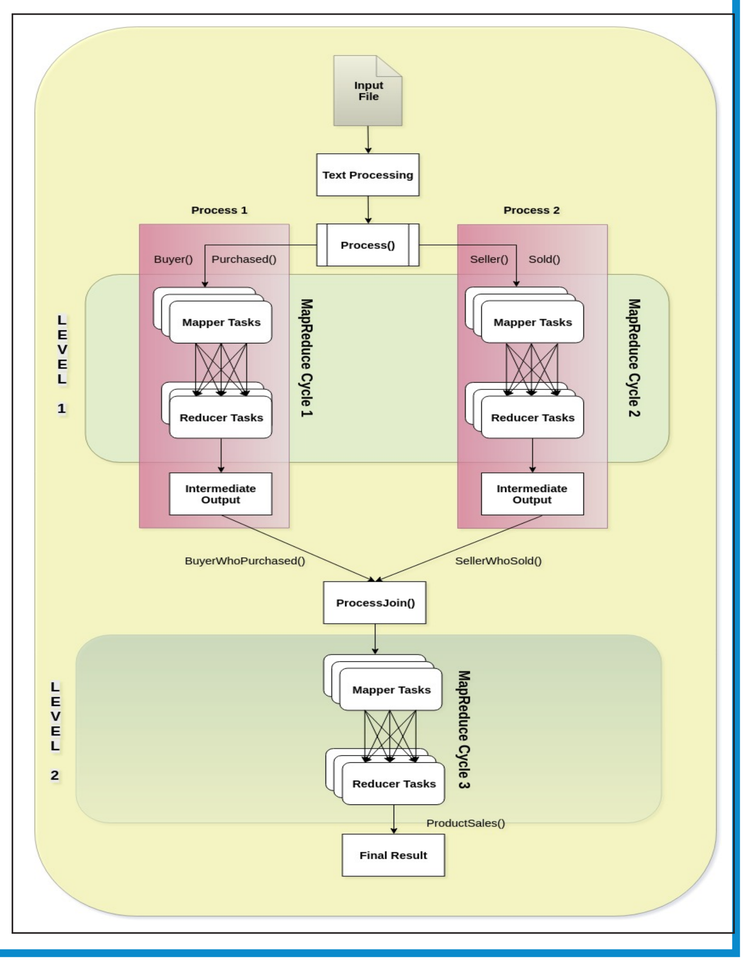
Original Design



Our Proposed Design

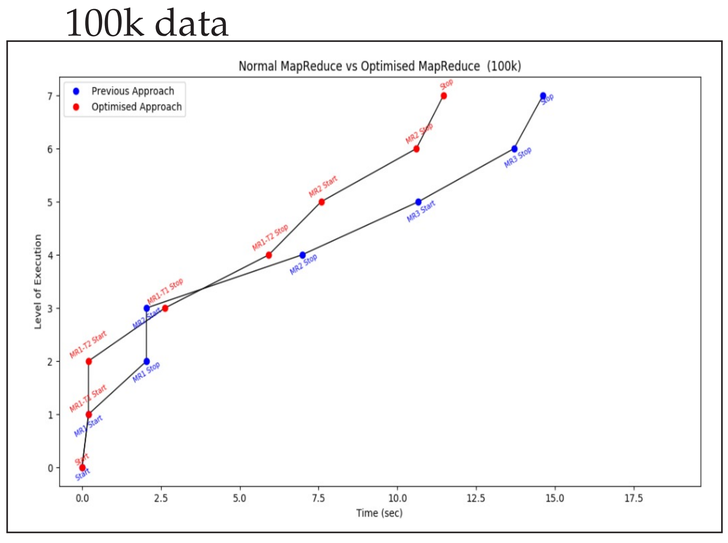


Flow Chart

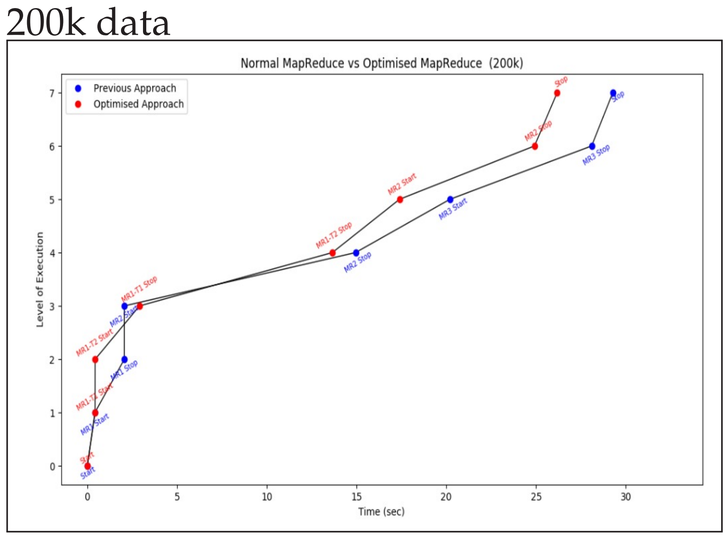


Evaluation/Observations

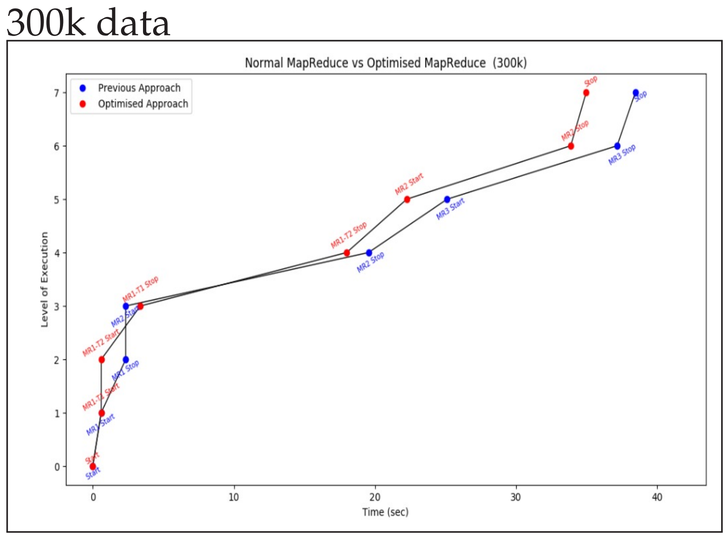
100k data



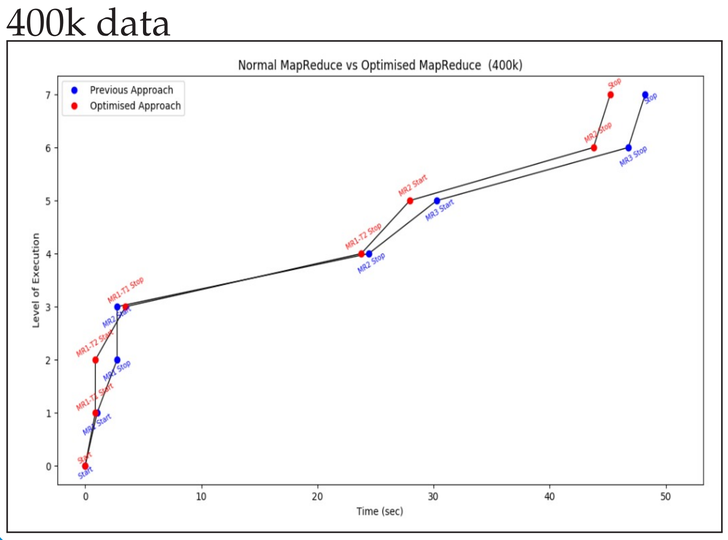
200k data



300k data



400k data



Future Work

We chose prolog queries to prove that our implementation of parallezing map reduce cycles is successful. Similarly, this concept can be extended to SQL and the like. Huge SQL queries can be processed to extract indenpendent operations and then these operations can be parallelized in a similar fashion.