Summary

Swift is a parallel programming language that allows writing scripts that helps to distribute program execution across distributed resources such as clouds, grids and HPC systems. Since the data is required for various applications and the data being too large the complexity of data management increases as the resources are required constantly when being requested by some application. Although the distributed and parallel computing resources can increase the speed of such computations but due to such complex nature of the distributed file systems the programming complexity increases. Therefore swift helps to reduce such complexity by helping the applications to efficiently utilize the parallel and distributed resources. Swift helps in simplifying the distributed parallel execution as it is a deterministic programming model. Most of the applications require a single message passing parallel program model others require large number of application invocations. Also scaling up requires the distribution of workloads among cores, processors or computers. Even if a large cluster is being provided to the users it may not be sufficient due to reasons such as congestion, limited bandwidth etc. Therefore it is required to use the available resources in hand without the need to reprogram.

There are various features that swift offers and have various advantages. 1) Swift is implicitly parallel and location independent 2) Does not replicate the capabilities instead calls the scripts as small applications. 3) Can execute scripts that perform high level of program invocations on parallel resources 4) swift helps users to specify process composition by representing processes as functions. Performance was also measured both on synthetic benchmark results and application performance results. According to the results the tasks which lasted 5 seconds, swift sustained 100 concurrent executions at a utilization rate of 85%. Second test was the ability of swift to measure tasks on large distributed memory systems. The system used was IBM Blue Gene /P. The observation was for a task of 100 seconds, swift achieves 95% utilization of 2048 compute nodes and for 30 seconds tasks it sustained an 80% utilization. Swift expect the input files to the functions to be in working directory of the application execution, and the output files are created in the working directory. Once the script of the swift is written it can be executed on any environment. The swift script is POSIX implementation. It provides encapsulation for program execution in distributed environment.

How is this work different from the related work ?

1. Swift coordinates the execution the execution of legacy applications which are coded into various programming applications. Thus they can be executed on heterogenous platforms. Other languages such as Linda, Strand and PCN are linked with the systems and support the composition of parallel functions.
2. The MapReduce programming model supports key-value pairs as input or output datasets and two types of computation functions, map and reduce; Swift provides a type system and allows the definition of complex data structures and arbitrary computational procedures.
3. In MapReduce, input and output data can be of several different formats, and new data sources can be defined although Swift provides a more flexible mapping mechanism to map between logical data structures and various physical representations.

4. MapReduce schedules computations within a cluster with a shared Google File System; Swift schedules across distributed grid sites that may span multiple administrative domains and deals with security and resource usage policy issues.
5. Swift tasks can be written in any programming language unlike Dryad which can only be written in C++
6. Swifts functional syntax is more natural .

Top 3 things the paper does well

1. Implicit parallelism : Swift is implicitly parallel and user does not need the knowledge of program execution location. Swift handles that as per the availability of the resources.
2. Support use of dynamic resources.
3. The execution environment of the swift inherently handles many things such as parallelism, task execution replication for reliability, load balancing, etc.

Things paper could do better

1. The collection data types only have 2 types array and structure. These seems to be very basic compared other languages. Many other collection types can be added to these.
2. The execution of the application can be guaranteed on any particular host, but is decided on the availability of the resources. Certain amount of control should be given to the application for the execution.
3. When the application invocation fails in swift, re attempt is again from resource selection and execution of the invocation. The execution time compromise which swift does in this has to properly analyzed

If you were to be an author of a follow up paper to this paper, what extensions would you make to improve on this paper?

The swift language is still at its early stages. Although swift is extremely fast and simple many aspects can be learned from other programming and scripting languages and can be implemented in swift. With the help of good comparison with other similar features from the other languages many better data types and user friendliness can be introduced to SWIFT.