Aakash Saini(2018CSM1001)

May 7, 2019

- **Observations:** Paper proposes the *Heron*, a modular architecture to provide the unparalleled performance at the large level for the real-time data centres at twitter. Heron's architecture is based *on micro kernel-based OS* and it provide a core set of services like scheduling and IPC. A stream of tweets, known as *spouts*, is submitted to Herons and **Topology Master** determines how many containers should be allocated and remaining containers run *Stream Manager, a Metrics Manager* and *a set of Heron Instances* which runs on JVM. Herons supports two layers in its architecture: **Data Processing Layer** which can run multiple tasks in different threads, and **Communication Layer** that makes all stream communication customized. Heron provides different modules in its API repository: **Resource Manager, Scheduler**and **State Manager**. Resource Manger implements basic functions of Heron Instances to containers, Scheduler interacts with the underlying scheduling framework such as **YARN**, and State Manager helps in distributed coordination and sorting of topology data. For the evaluation Herons experiments were performed on Microsoft HDIn-sight. Each machine has one 8-core Intel Xeon E5-2673CPU@2.40GHz and 28GB of RAM. Experiment demonstrates that although Heron employs an extensible and general-purpose architecture, it can significantly outperform Storm's more specialized architecture.

- **Conclusions:** Heron's modular and extensible architecture helps it efficient for various applications and attains the flexibility without sacrificing the performance. Self-contained modules operating at the top of kernel makes Heron to run it on the different underlying software stacks. It simplifies the detection of performance problems as well as tuning and maintenance of overall software stack.

- **Limitations:** Potential overhead during the data transfer between the different modules of system is more. After reaching at the time threshold, additional queuing cause the performance degradation. Throughput decreases with the high degree of parallelism because end-to-end latency increases.

- **Future work:** In future it can be made compatible to the monolithic kernel also. Overheads caused due to the data reading and writing to the external services can be minimised by implementing multi-thread based communications.