Students,
Here is the article on twitter I mentioned last Wednesday:

https://blog.twitter.com/2013/new-tweets-per-second-record-and-how

Questions:
1. What did they redesign?

The architecture of the twitter subsystems (used to manage how tweets are received and written to database-storage) and the full software-lifecycle for twitter was redesigned.

2. Why?

The architecture was redesigned so it could accommodate much higher concurrent interaction on their website and so multiple engineering teams could take on parts of the software individually, without having to understand the architecture in its entirety, based on a large, tightly-coupled codebase, so as to cope with very high levels of concurrency as observed for example during the 2010 World Cup and the worldwide New Year's Eve celebrations.

3. Has it worked?

The twitter engineering team's blog post claims their redesign has functioned very well, including affording their engineering teams efforts to be paralellized and allowing development and implementation of new features that afforded them an enhanced auto-indexing capability superior to that for which MySQL's indexing functionality was designed and a feature that allows deploying new features to a limited set of live users (similar to how google operates, of course,also) whilst maintaining four 9's uptime.

4. How do they know it worked?

They have measured their success with metrics for reliability, performance capabilitiy (from 200-300 reqs/sec/host to 10-20k), and comparatively assessed the limitations the were overcome by switching to a Ruby-based VM to a JVM in terms of processes and threads at a kernel-level and how they influence number of tweets that can be 'handled' in comparison to networking, processor, IO and other latencies.