livelessons
video instruction from technology experts

Lesson 4: Spark Internals

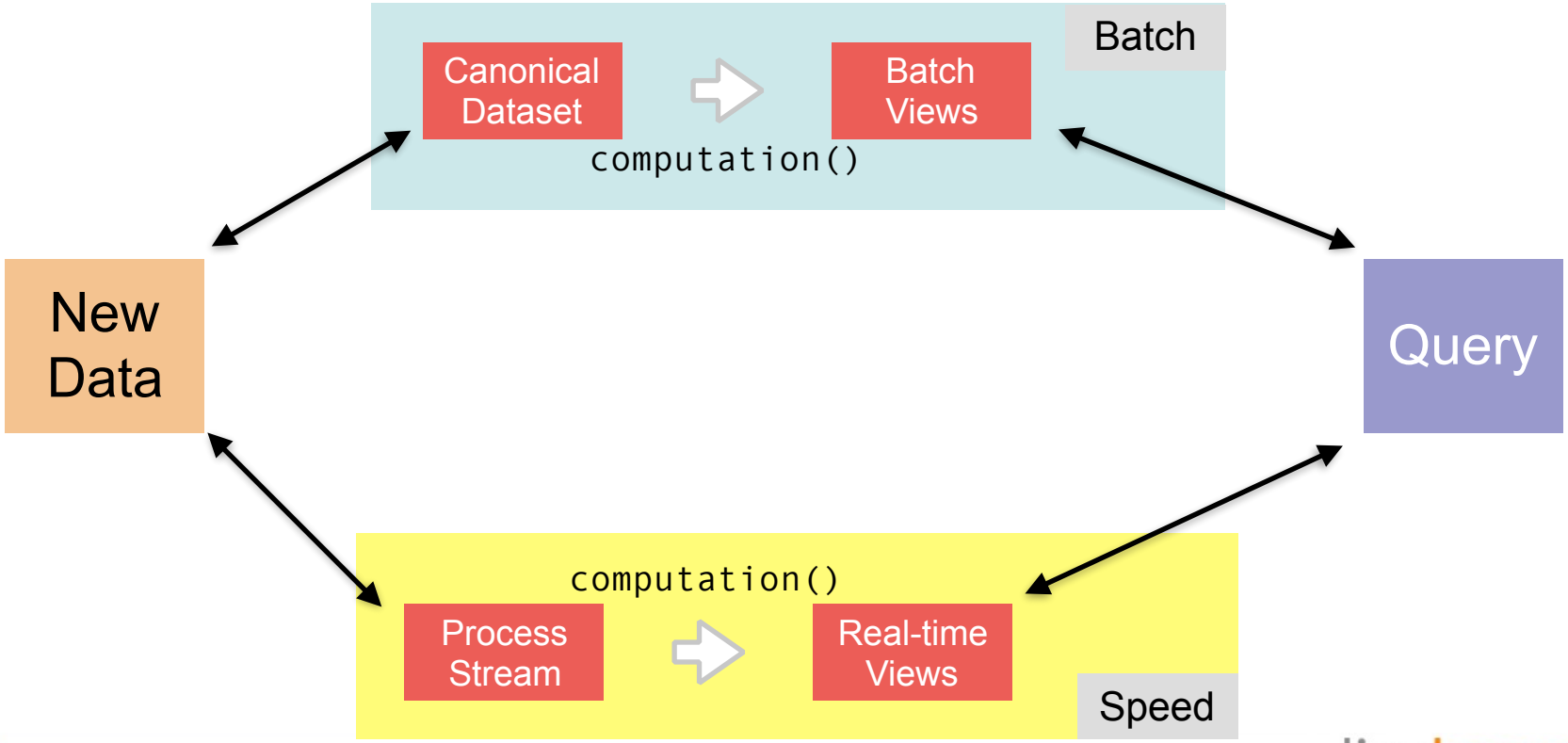
4.2 Building Systems that Scale

CAP Theorem

- **Consistency:** all nodes see the same data at the same time
- **Availability:** every request receives a response about whether it succeeded or failed
- **Partition tolerance:** the system continues to operate despite arbitrary partitioning due to network failures

Choose two: except you can't really sacrifice P!

Lambda Architecture



So, why the excitement about the Lambda Architecture? I think the reason is because people increasingly need to build complex, low-latency processing systems. What they have at their disposal are two things that don't quite solve their problem: a scalable high-latency batch system that can process historical data and a low-latency stream processing system that can't reprocess results. By duct taping these two things together, they can actually build a working solution.

- Jay Kreps

Review

- Different distributed architectures have different tradeoffs
- Scaling up is logically easier for programmers, and easier to reason about.
- Scaling out is theoretically unbounded, but coordinating and reasoning in a distributed system is hard
- You have to make compromises to ensure fault tolerance in a distributed system, though there are ways to mitigate these.

Next Up: Spark Execution Context