Messaging API - methods or resources

Is message order important?

Some messaging services, ActiveMQ, for example, preserve the order of messages from sender to receiver, although messages from multiple senders to the same receiver will be interleaved. In these systems some component must gather and sort the messages and release them one-at-a-time to each receiver. A network or host outage on one component might delay all messages.

Other messaging services, such as AWS's SNS/SQS, do not preserve order. Messages read by a receiver might not appear in the order they were sent. In these systems any members of a distributed cluster can send whatever messages it has for a particular client. Network or host outages affect only the messages contained solely on the affected host.

Ordered systems are in many cases easier to use, but less scalable and reliable.

Methods or Resources?

If the messages are carrying 'methods', for instance:

1. at 10am: set a = a + 4
2. at 11am: set a = a * 10

They clearly must be processed in order. A method requires a particular state of a resource. The order may provided by the service or locally, by the listener. In the latter case one might collect messages for some period of time and then process them according to a message timestamp. This can negate most of the advantages of the unordered service. There can be only one receiver per queue; there can likely be only one sender per queue.

If instead the messages carry resource representations, i.e., what one might receive from a REST GET, order may not be important at all. If a message contains:

1. at 11am: a is 20

one compares the time of the last modification of **a** with *11am*. If earlier, set 'a=20'. If later, ignore the message.

Messages that carry resource representations are far more flexible, scalable and reliable. And, I might add, easier to implement in systems of heterogeneous components.