While both Service-Oriented Architecture (SOA) and Microservices Architecture share the common goal of breaking down applications into smaller, independent components, they differ significantly in their approach, scope, and implementation.[1] Here's a breakdown of the key differences:

1. Granularity and Scope:

- **SOA:**
  - Focuses on enterprise-level services, often representing large business functions.
  - Services are typically larger and more coarse-grained.
  - Often involves a centralized Enterprise Service Bus (ESB) for communication.
- **Microservices:**
  - Focuses on fine-grained, independent services that represent specific business capabilities.[2]
  - Services are much smaller and more granular.[3]
  - Emphasizes decentralized control and autonomous teams.[4]

2. Communication and Integration:

- **SOA:**
  - Often relies on a centralized ESB for communication between services.[5]
  - Can use various protocols, including SOAP and REST.[6]
  - Tends to have more complex integration patterns.
- **Microservices:**
  - Prefers decentralized communication, often using lightweight protocols like REST or gRPC.[7]
  - Emphasizes API-driven communication.[8]
  - Promotes simpler, more agile integration.[9]

3. Data Management:

- **SOA:**
  - May involve shared databases or centralized data management.[10]
  - Can lead to data coupling between services.
- **Microservices:**
  - Advocates for decentralized data management, where each service owns its own data.[11]
  - Minimizes data coupling and promotes data independence.

4. Deployment and Scalability:

- **SOA:**
  - Services are often deployed as monolithic applications.
  - Scalability can be complex due to shared infrastructure.
- **Microservices:**
  - Services are deployed independently, allowing for flexible scaling.[12]
  - Enables independent deployment and updates.[13]

5. Technology and Implementation:

- **SOA:**
  - Can be implemented using various technologies and platforms.[14]
  - Historically associated with more heavyweight technologies like SOAP and XML.[15]
- **Microservices:**
  - Often relies on lightweight technologies like containers (Docker) and orchestration platforms (Kubernetes).[16]
  - Emphasizes automation and continuous delivery.[17]

In essence:

- SOA aims to achieve interoperability between enterprise-level services, often focusing on larger, more complex business processes.[18]
- Microservices, on the other hand, focus on creating highly scalable and agile applications by breaking them down into small, independent, and autonomous services.[19]

Here's a simplified table:

Feature	**SOA**	**Microservices**
Granularity	Coarse-grained	Fine-grained
Communication	ESB, various protocols	API-driven, lightweight protocols
Data	Shared databases	Decentralized data
Deployment	Monolithic	Independent
Scaling	Complex	Flexible

Essentially, Microservices are often thought of as an evolution of SOA, taking the concept of service-oriented architecture, and greatly increasing the granularity, and level of independence of the individual services.[20]