Orleans: Virtual Actors for Programmability and Scalability of Cloud Applications

Philip A. Bernstein, Sergey Bykov, Alan Geller, Gabriel Kliot, Jorgen Thelin
Microsoft Corporation

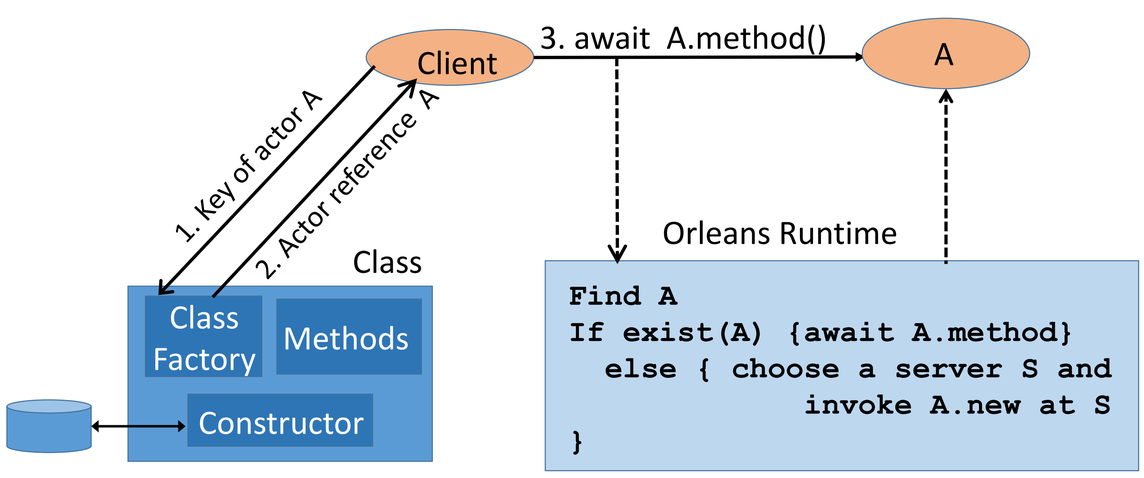
Problem

- Building interactive, distributed services is hard
  - Challenges: scalability, reliability, concurrency, …
- The actor model helps make it easier
  - Actors are objects that do not share state
  - E.g., game, device, phone, scoreboard, region
- But problems remain
  - Actor lifecycle management, actor failure & recovery, actor placement, distributed races, resource management

Novel Solution: Virtual Actors

- Always exist
- Cannot be created or destroyed
- Are location transparent
- Are instantiated when referenced
- Are reclaimed when not used
- Are analogous to virtual memory

How to Invoke Virtual Actor "A"



Implementation – Project Orleans

- Define .NET classes in C#, as if they run in one process
- Orleans scales out the app on a cluster of servers
- ActorID→Server mapping is stored in a distributed hash table and cached locally.
- If server S fails, actors at S are reactivated elsewhere
  - Kill & reactivate actors mapped by S's directory
- Single-threaded actors, communicating via async RPC
- Timer services, load balancing, declarative persistence, …

Orleans Benefits

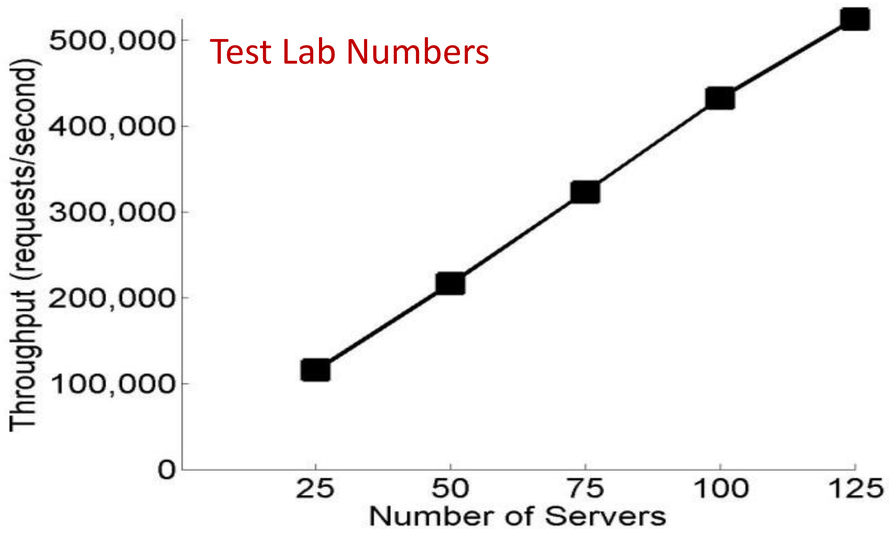
- Helps desktop developers succeed with distributed apps
  - Ensures app is scalable, reliable, and elastic by default
- Improves productivity of distributed systems experts
  - 3-5x less code to write than on a bare VM
- Performance is near-real-time (milliseconds)

Adoption

- Used extensively by Microsoft and its customers
  - For all cloud services of the Xbox game Halo 4
  - Telemetry, Internet of things, and many other games
- Microsoft Azure's Reliable Actors is based on Orleans' API
- Project "Orbit " is a Java implementation of Orleans-style virtual actors by BioWare, a division of Electronic Arts
- Thriving open source community at GitHub

Scalability

- Near linear scaling to 100K's requests/second
- Scalable in number of actors
- Multiplexes network connections, threads, and memory buffers for efficiency
- Location transparency simplifies scaling up or down
- Elastic – transparently adjusts to adding or removing servers



Request: Client → Actor 1 → Actor 2

Orleans is open source at https://github.com/dotnet/orleans