Orleans: Distributed Virtual Actors for Programmability and Scalability

Papers We Love #12
San Francisco
February 19th 2015
Caitie McCaffrey
Distributed Systems Engineer

@Caitie
CaitieM.com
Orleans: Distributed Virtual Actors for Programmability and Scalability

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

eXtreme Computing Group MSR
Orleans: Cloud Computing for Everyone
November 30th 2010

Orleans: a Framework for Cloud Computing
October 2011

Orleans: Distributed Virtual Actors for Programmability & Scalability
March 2014
The Actor Model

A framework & basis for reasoning about concurrency

A Universal Modular Actor Formalism for Artificial Intelligence
Carl Hewitt, Peter Bishop, Richard Steiger (1973)
Actor Actions

• Send a Message
• Create New Actors
• Designate the Behavior to be used on the next Message
Orleans

“Orleans is a runtime and programming model for building distributed systems, based on the actor model”
Virtual Actors

“An Orleans actor always exists, virtually. It cannot be explicitly created or destroyed”
Virtual Actors

• Perpetual Existence
• Automatic Instantiation
• Location Transparency
• Automatic Scale out
Runtime

• Messaging
• Hosting
• Execution
Programming Model

- .NET Framework
- Actor Interfaces
- Promises
  - Actor References
- Turns
- Persistence
namespace HelloWorldInterfaces
{
    /// <summary>
    /// Orleans grain communication interface IHello
    /// </summary>
    public interface IHello : Orleans.IGrain
    {
        Task<string> SayHello();

        Task<string> SayGoodbye();
    }
}
Programming Model

- .NET Framework
- Actor Interfaces
- Promises
- **Actor References**
- Turns
- Persistence
IHello friend = HelloWorldInterfaces.HelloFactory.GetGrain(0);
string message = await friend.SayHello();
Programming Model

- .NET Framework
- Actor Interfaces
- Promises
- Actor References
- Turns
- Persistence
public class HelloGrain : Orleans.GrainBase, HelloWorldInterfaces.IHello
{
    int sayHelloCount = 0;
    int sayGoodbyeCount = 0;

    Task<string> HelloWorldInterfaces.IHello.SayHello()
    {
        sayHelloCount++;
        return Task.FromResult(" I say: Hello! " + DateTime.UtcNow.ToDateTimeString());
    }

    Task<string> HelloWorldInterfaces.IHello.SayGoodbye()
    {
        sayGoodbyeCount++;
        return Task.FromResult("I say: Goodbye! " + DateTime.UtcNow.ToDateTimeString());
    }
}
Reliability

“Orleans manages all aspects of reliability automatically”
Isolation
Messaging Guarantees
Let’s Talk About CAP
Orleans is AP
Halo 4: Statistics Service
Halo 4: Presence Service
Performance & Scalability
“Orleans applications run at very high CPU Utilization. We have run load tests with full saturation of 25 servers for many days at 90%+ CPU utilization without any instability”
Figure 6: Throughput of Halo 4 Presence service. Linear scalability as number of server increases.
Figure 7: Throughput of Halo 4 Presence service. Linear scalability as number of actors increases.
Conclusion

- Virtual Actors
- AP Actor Activation
- Developer Productivity
Get Orleans

https://github.com/dotnet/orleans
Discussion