# Introduction

Scalability

Transaction models and theory behind them

Message based architectures

# Performance and Scalability

Performance vs scalability

Vertical vs horizontal

*Vertical scalability graph – response time vs concurrent users*

*Horizontal scalability graph – resources / cost vs concurrent users*

# ACID Transaction Model

Most common transaction model, all major relational databases use ACID

Generally what we are talking about when we say ‘consistency’.

ACID

Locking

* Isolation levels
* Escalation
* Deadlocks

Scalability

# CAP Theorem

# BASE Transaction Model

Eventual Consistency

* Staleness
* Concurrency and staleness

# Message-Based Architectures

Loose coupling

* Kind of “inversion of control”
* compare to “traditional” architectures
  + components depend explicitly on other components and need to call out to them – coupled to the API
  + **Example / diagram**
  + Real advantage is to be able to add and remove subscribers without to publisher knowing / caring
* as subscribers grow, remember SRP

Guaranteed delivery

Error handling / compensation

* asd

Transactions / idempotence

* Transactional queues generally require distributed transactions (MSDTC) – undesirable
* (idempotent) “unchanged in value following multiplication by itself; "this matrix is idempotent” - <http://wordnetweb.princeton.edu/perl/webwn?s=idempotent>
* “Idempotence is the property of certain operations in [mathematics](http://en.wikipedia.org/wiki/Mathematics) and [computer science](http://en.wikipedia.org/wiki/Computer_science), that can be applied multiple times without changing the result beyond the initial application” - <http://en.wikipedia.org/wiki/Idempotence>

# CQRS

Separate read / write models

ACID

# Composite UIs

Loose coupling!

# SOA

# Tribe

Intro

* why?
* focused around reducing friction
* Pure JavaScript
  + run the same code on both the client and server
  + such code needs to be constrained and given a common environment
  + highly testable!
* Ultimate debugging experience
* Simple and powerful deployment
  + Purely static resources
  + CDN
  + Caching

Components

* PubSub
* Composite
  + Dependencies
  + Forms
  + Components
* Server side
  + Previously SignalR, now nodejs based
  + Pure JavaScript
  + Dependencies
  + PackScript

# Learn More