Developers and Architects

Strategies 2017

Oliver Sturm • @olivers • oliver@oliversturm.com

Oliver Sturm

- Training Director at DevExpress
- Consultant, trainer, author, software architect and developer for over 25 years
- Microsoft C# MVP
- Contact: oliver@oliversturm.com

Agenda

Idea: Talk about technology

- Application building blocks
- Services
- Microservices
- Data persistence
- User Interfaces
- Programming Languages
- Mobile
- Cloud
- Open Source

Application Building Blocks

- What is an “application” made of?
- Terminology check:
    - Client application
    - Server application
    - Web application
    - Application system
    - Enterprise application

Building Blocks

Terminology: Anwendung, Programm

- Manche sehen ein Programm als eine unfertige Anwendung
    - Fehler, etc…

Terminology: Client Application

- Arbeitet “lokal” auf einem System
- Evtl. Interaktion mit Anwender
- Teilnahme in einem Client/Server-System

Terminology: Server Application

- Liefert Informationen an Clients
- Evtl. indirekte Arbeit fuer Clients
- Nicht jede Anwendung, die auf einem “Server” laeuft, ist eine Serveranwendung
- “Service” kann eine Serveranwendung sein, muss aber nicht

Terminology: Web Application

- Verwendung von “Web-Technologien”
- Traditionell, serverseitige Generierung von Web-Inhalten
- Aktuell: Ausfuehrung im Browser
- Achtung: viele solche Anwendungen koennten auch als Client-Anwendungen bezeichnet werden

Terminology: Application System

Terminology: Enterprise Application

Services

- Part of most architectural concepts
- SOA?
- Web Services
- “Real-time web?” SignalR? socket.io?

Services – SOA

Remember the four tenets Don Box got excited about?

- Boundaries are explicit
- Services are autonomous
- Services share schema and contract, not class
- Service compatibility is determined based on policy

SOA *resulted* in a very formal understanding of service architecture, which is fortunately not shared by too many architects today.

Web Services

- ASMX – WSE – WCF – WSDL – SOAP – Microsoft’s world of enormous complexity intended to solve a very simple problem
- RESTful services: the most complicated part is the name
    - URLs and HTTP methods
    - JSON, XML and possibly other data formats, using content negotiation

Services – Real-time Web

- WebSockets and their various ancestors
- Bi-directional communication

Reasoning for real-time web techniques:

- Serverseitige unaufgeforderte Benachrichtigungen
- Nicht als Ersatz fuer AJAX zu betrachten

Microservices

How big is a microservice? It depends.

- Do one “thing” well. What’s a "thing"? It depends.
- Two-pizza team
- Throwawayable
- Focus on boundaries and business context, not on lines of code

Microservices – Communication

- Direct communication between services
- Message Queues
- Service Bus (ESB)

Microservices – Composition

- Function level: AWS Lambda, Azure Functions – “Serverless” Computing
    - Integration with cloud infrastructure for triggering and output generation
- Docker containers
- docker-compose
- Cloud container services (ecs-cli, Azure Docker VM extension)
    - Also support composition

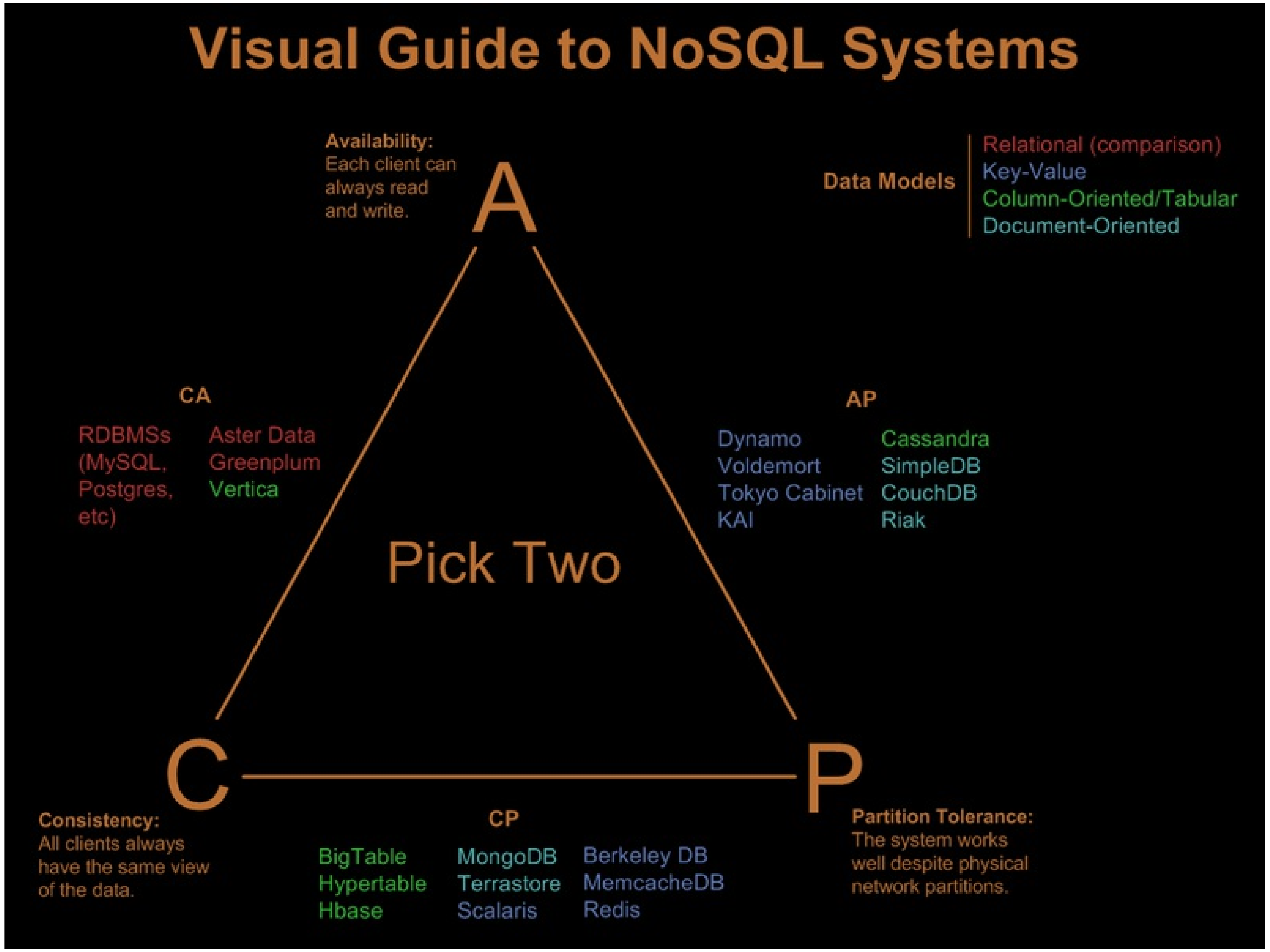
Microservices – Reasoning

- Grosse Flexibilitaet
- Erzwungene Modularisierung
- Langfristige Pflegbarkeit

Data Persistence

- Relational databases
- NoSQL options
    - Key/value and column family stores
    - Document
    - Data analytics (e.g. MapReduce)

Data Persistence – NoSQL



The only image in this presentation, used with permission from Nathan Hurst, nathan@developersforgood.org

http://blog.nahurst.com/visual-guide-to-nosql-systems

Eventual Consistency

http://queue.acm.org/detail.cfm?id=2462076

Reasoning NoSQL vs RDBMS:

Data Persistence – ORM

- Choice of frameworks
- Top Down or Bottom Up?
- DB Independence

Reasoning:

Data Persistence – CQRS

Command/Query Responsibility Segregation

- Separate query and command models
- Conflicts with ORM?
- Event Sourcing
    - Eventual consistency

Reasoning CQRS and Event Sourcing:

User Interfaces

- Platforms
    - Native: WinForms, XAML
    - HTML
        - Electron

Reasoning for native UI platforms:

UI Application Patterns

- MVVM
- Flux

HTML UI – Where to Render

- Traditional web-server based rendering?

Reasoning:

Programming Languages

- .NET: C#, VB.NET, F#, others?
- JavaScript: Native, TypeScript, CoffeeScript, LiveScript, others?

Mobile

- Mobile support as a conceptual module
- Strategic platform?

“Native” Mobile

- iOS SDK
- Android SDK
- Windows Phone?

Reasoning:

Mobile .NET

- Xamarin
  - Native
  - Forms

Reasoning:

Mobile – HTML/Hybrid

- HTML (5), JavaScript, CSS
- PhoneGap/Cordova, CrossWalk, nw.js, …
- Cross-platform

Reasoning:

Cloud

- Deployment option
    - Related: Docker?
- Managed infrastructure

Cloud functionality

- Supplied services, vertical features
- Base platform functionality
    - Dynamic scalability
    - SLA
- Serverless computing

Cloud – Legal Considerations

- Locations
- Industry/governmental requirements

Cloud Options

- Azure, Amazon Web Services (PaaS, IaaS)
- PaaS: Google (also some IaaS now), Heroku, others
- SaaS: Office 365, Azure/AWS Websites, …

Cloud Reasoning

- For/against cloud:
- For/against specific platforms, IaaS, PaaS:

Open Source

- Everybody does it, right?
- Give and take…

Reasoning:

Sources

- This presentation:
  - https://oliversturm.github.io/developers-and-architects/basta-2017
  - Deprettified content in pdf format: https://oliversturm.github.io/developers-and-architects/basta-2017/slidecontent.pdf

Thank You

Please feel free to contact me about the content anytime.

oliver@oliversturm.com