Developers and Architects

Strategies 2018

Oliver Sturm • @olivers • oliver@oliversturm.com

DevExpress®

Oliver Sturm

- Training Director at DevExpress
- Consultant, trainer, author, software architect and developer for over 25 years
- 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: Client Application

Terminology: Server Application

Terminology: Web Application

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:

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

- Manual composition? Painful.
- Docker containers
- docker-compose
- Cloud container services (ecs-cli, Azure Docker VM extension)
    - Also support composition

Microservices — Serverless

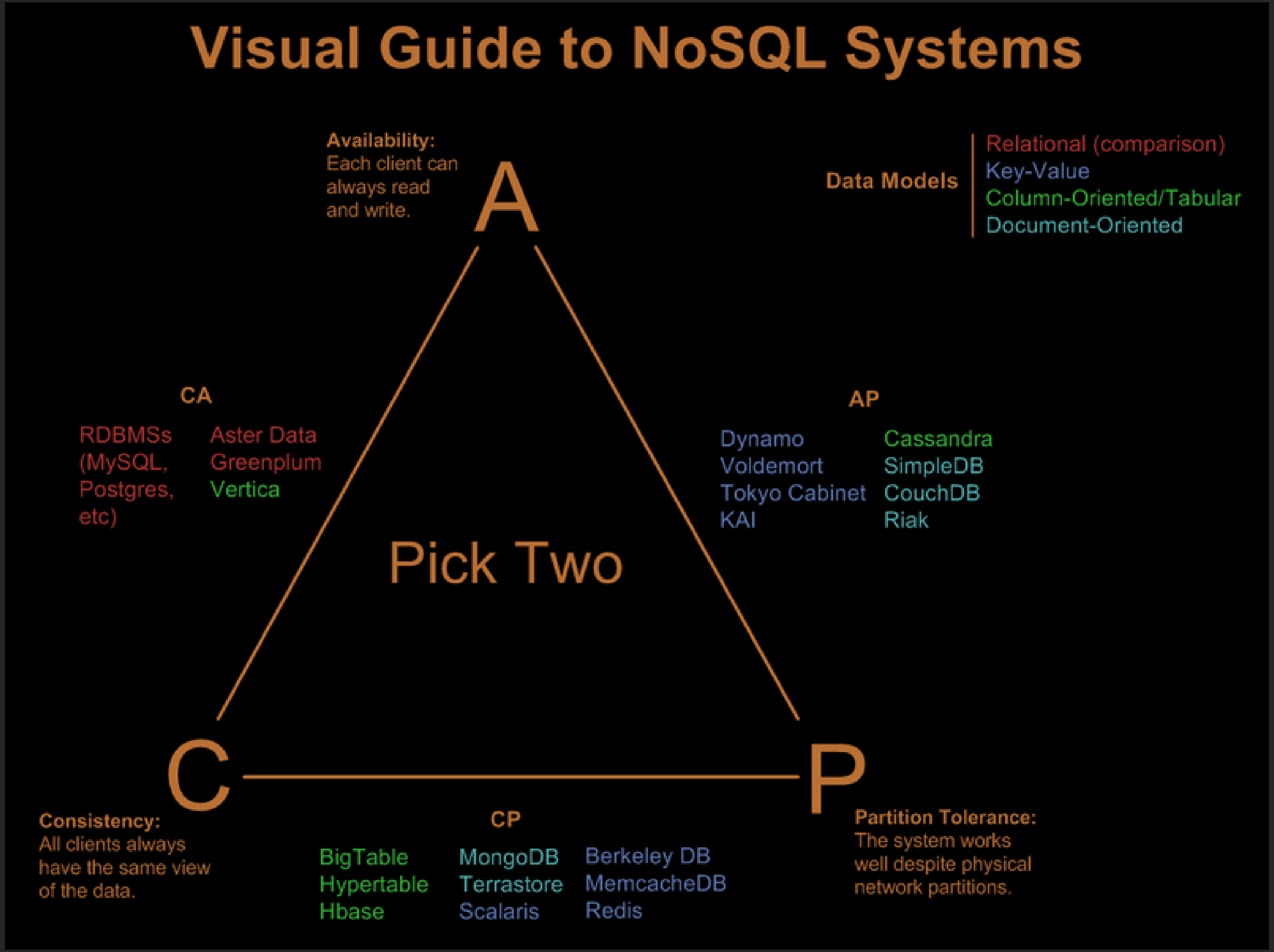
- Function level composition: AWS Lambda, Azure Functions, Google Cloud Functions, ...
    - Integration with cloud infrastructure for triggering and output generation
- Event driven, scaleable systems with minimal infrastructure management requirements
- Pay as you go
- Lock-in effect
- Debugging, Testing...

Microservices — Reasoning

Data Persistence

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

Data Persistence — NoSQL



Visual Guide to NoSQL Systems

Availability:
Each client can always read and write.

Data Models
- Relational (comparison)
- Key-Value
- Column-Oriented/Tabular
- Document-Oriented

CA

RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica

AP

Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak

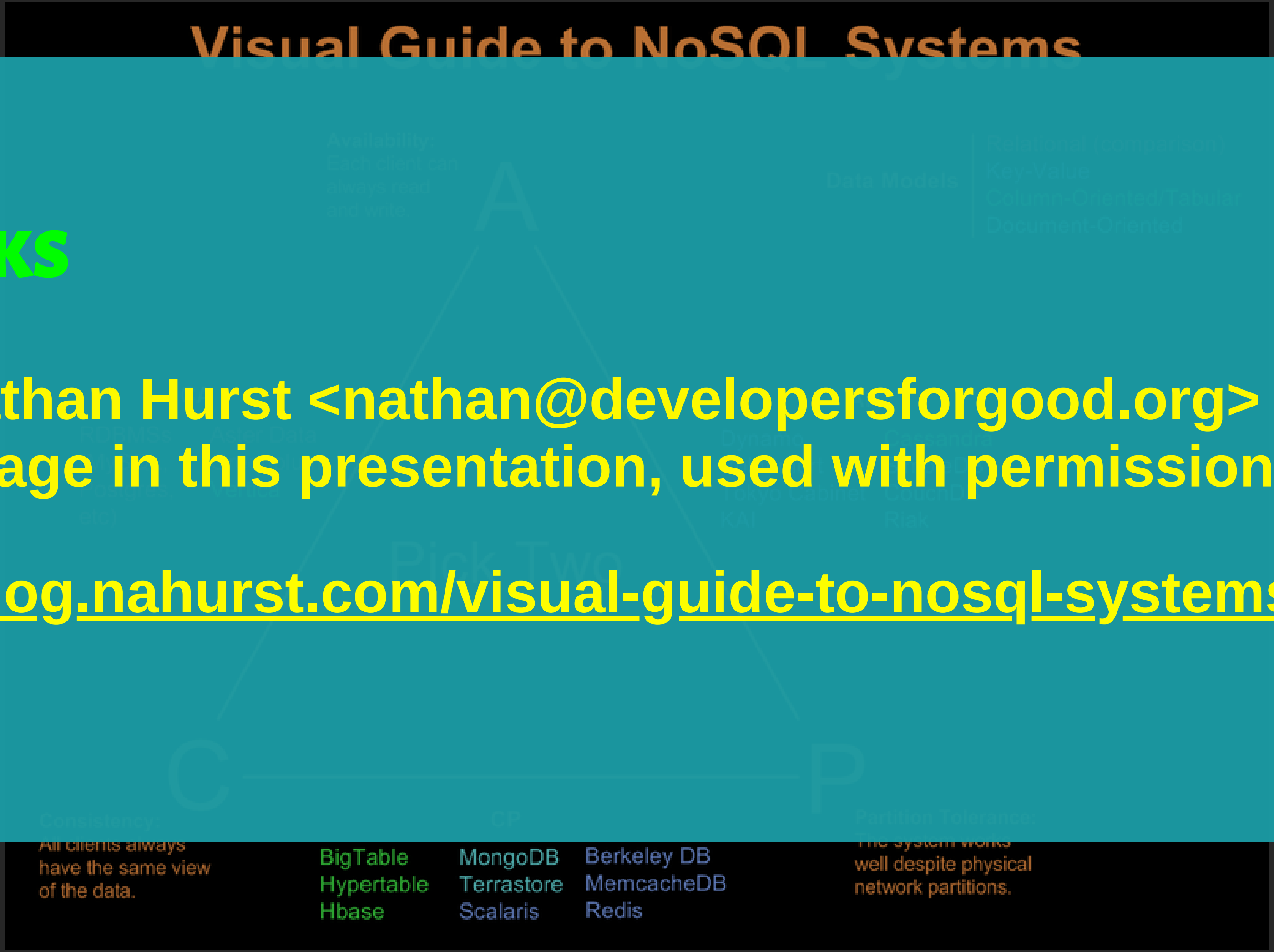
CP

BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis

Consistency:
All clients always have the same view of the data.

Partition Tolerance:
The system works well despite physical network partitions.

Data Persistence — NoSQL



Thanks

... to Nathan Hurst <nathan@developersforgood.org> for the only image in this presentation, used with permission.

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

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, Mobile
  - 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, related orchestration (Kubernetes, ...)?
- 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, Google (PaaS, IaaS, FaaS, ...)
- PaaS: 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/template
  - PDF download: https://oliversturm.github.io/developers-and-architects/template/slides.pdf

Thank You

Please feel free to contact me about the content anytime.

Oliver Sturm • @olivers • oliver@oliversturm.com

DevExpress®