Návrh architektury škálovatelné cloudové služby aneb nespalte se v cloudu

Mgr. David Gešvindr
MCSE: Data Platform | MCT | MSP
david@wug.cz

Motto

- **Běžná webová aplikace navržená pro provoz na vlastním serveru není schopná využít potenciálu cloudu**

- Její provoz bude drahý a neefektivní
- Aplikace je třeba navrhovat s ohledem na vlastnosti cloudového prostředí
  - Široké portfolio dostupných služeb
  - Kdykoliv může cokoliv selhat
  - Třeba optimalizovat podle více kritérií: cena, škálovatelnost
    - Vysoké náklady na výkonnou relační databázi

Osnova

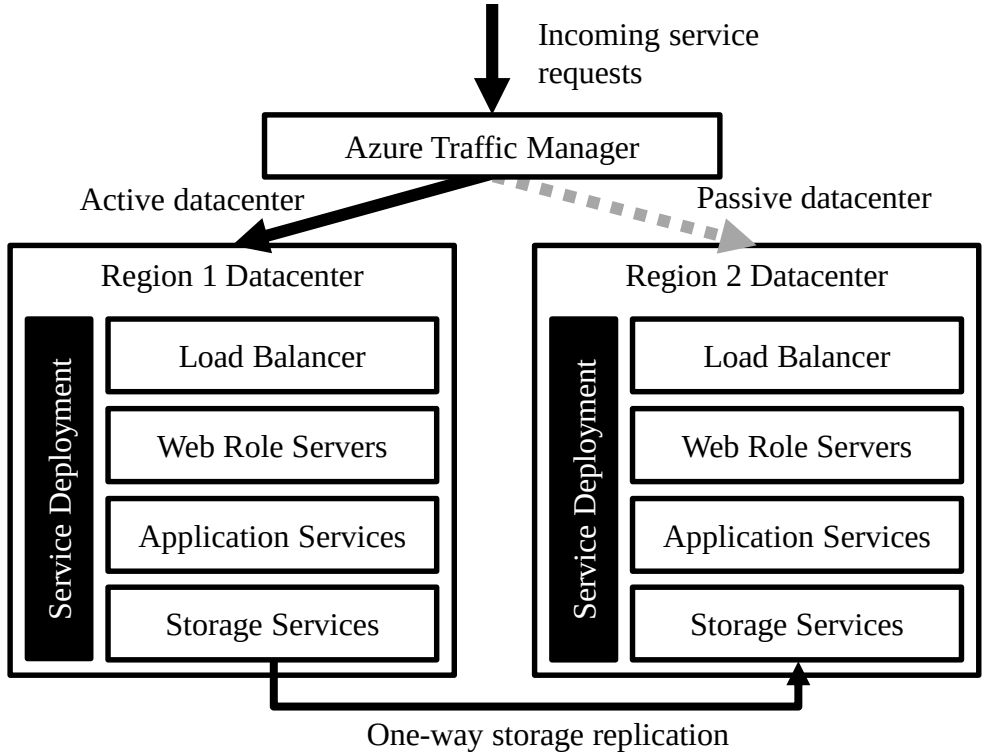
1. Metriky kvality
2. Case-studies
3. Taktiky aplikovatelné pro návrh SW architektury cloudové služby

Osnova

1. **Metriky kvality**
2. Case-studies
3. Taktiky aplikovatelné pro návrh SW architektury cloudové služby

Dostupnost

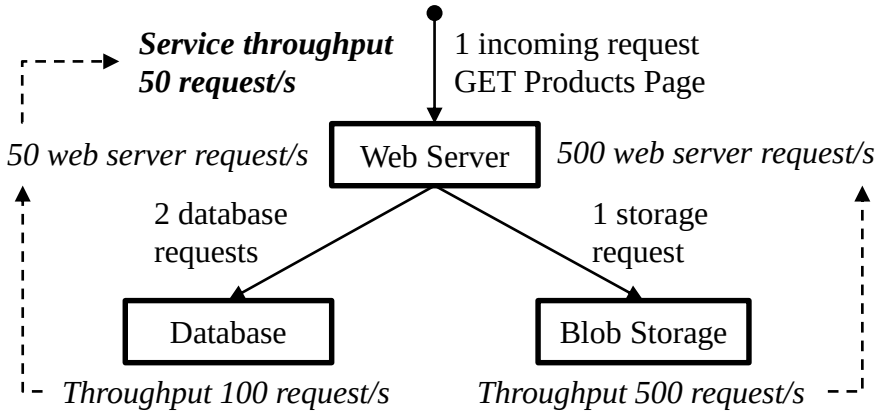
- Platforma disponuje výbornou dostupností
- Dočasné výpadky
  - Implementace ochrany proti Transient Errors
- Selhání uživatele – poškození dat, recovery
- Výpadek datacentra
  - Distribuce uživatelů mezi datacentry



Propustnost

- **Počet požadavků, které aplikace musí zpracovat za jednotku času**

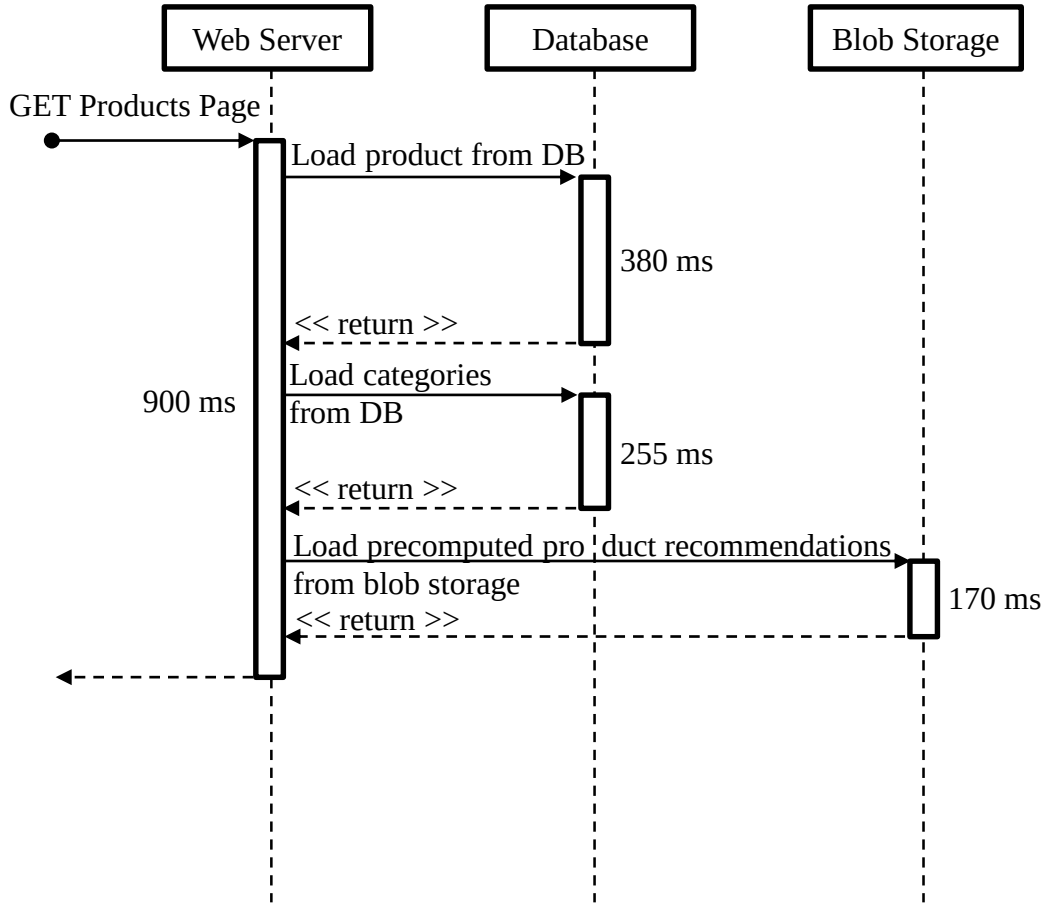
- Je silně závislý na jednotlivých komponentách, které se účastní zpracování požadavku
- Je třeba zavčasu identifikovat slabé místo



- Pozor na rozdíl mezi **průměrnou propustností** a **špičkovou propustností**

Odezva

- Zpoždění při zpracování operace
- Je dána
  - Komunikační latencí
  - Latencí jednotlivých komponent



Škálovatelnost

- Škálovatelnost charakterizuje, jak dobře bude řešení pracovat, pokud zvětšíme **problém**
- Neplést s:
  - **Výkonnostními atributy** – popisují chování při daném množství zdrojů
  - **Elasticitou** – Popisuje, jak rychle a přesně systém zareaguje alokací a dealokací zdrojů, aby se přizpůsobil potřebám aplikace

- **Pokud aplikace není škálovatelná, navýšení prostředků nepovede k výraznému zlepšení výkonnostních atributů**

Náklady na provoz

- Je nutné uvážit, jaké jsou skutečné náklady na provoz dané služby v cloudu

- Problém:
  - Platíme za využití služeb, náročné predikovat skutečnou úroveň využití

Náklady na vývoj

- Podobné služby nabízí odlišnou funkcionalitu a jiné výkonnostní charakteristiky
- Podle nabízené funkcionality se také liší pracnost jejich zakomponování do architektury aplikace

- Například:
  - Azure Storage x Azure SQL Database

Problémy při návrhu

- **Zmíněné metriky kvality jsou velmi často konfliktní**

- Zlepšení jedné skupiny metrik vede ke zhoršení jiných metrik

Osnova

1. Metriky kvality
2. **Case-studies**
3. Taktiky aplikovatelné pro návrh SW architektury cloudové služby

Case study: Volby v USA

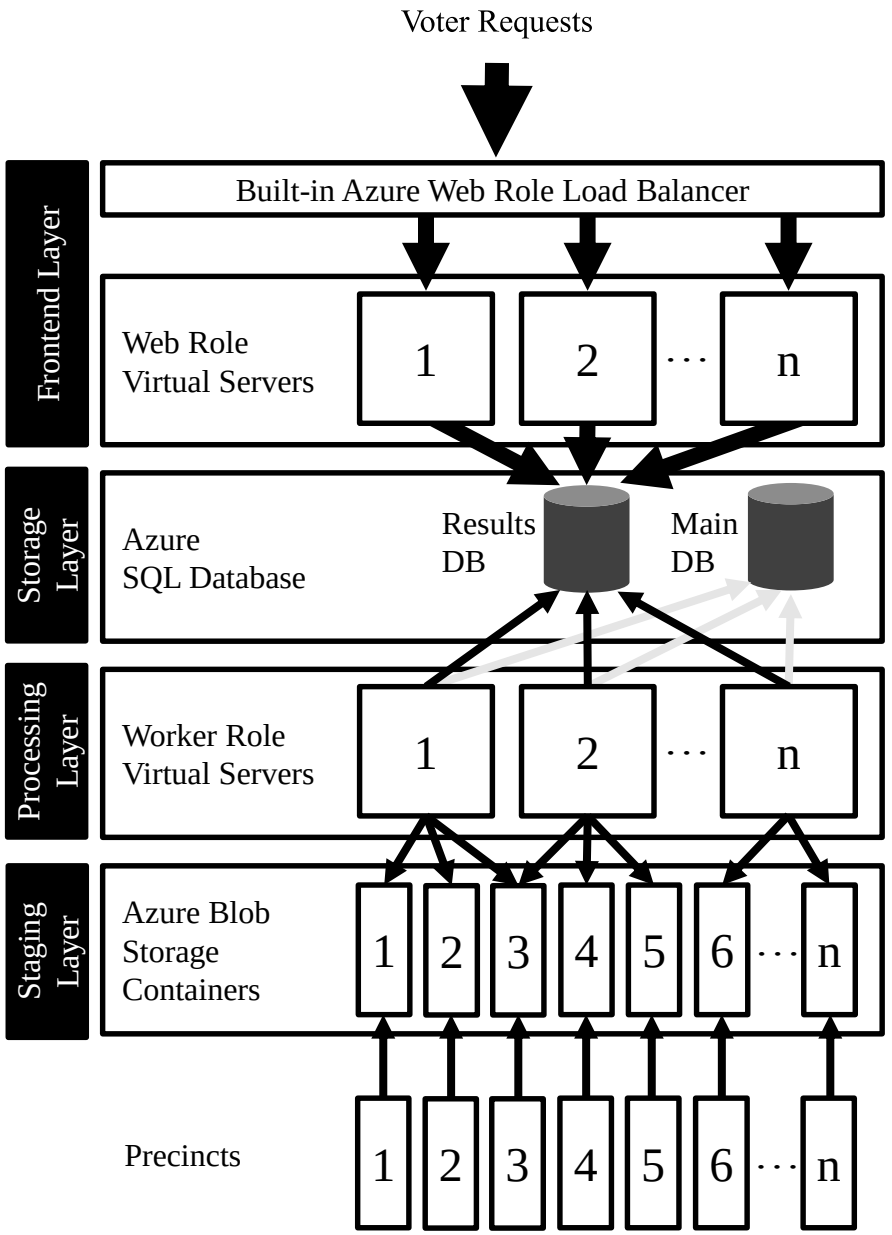
- Příklad prezentován na konferenci TechEd 2014 technickým ředitelem Azure **Markem Russinovichem**

- Celá přednáška: http://channel9.msdn.com/Events/TechEd/Europe/2014/CDP-B337

- **Sledování výsledků voleb**

Architektura služby

- Volební výsledky se nahrávají do Storage
- Worker role je průběžně zpracovávají
- Výsledky ukládají do relační databáze s výsledky
- Webové servery načítají výsledky z databáze



Očekávaná zátěž

- Ukázalo se, že každý požadavek na zobrazení stránky s výsledky **vyústí v 10 dotazů do databáze**
- Očekávaná zátěž

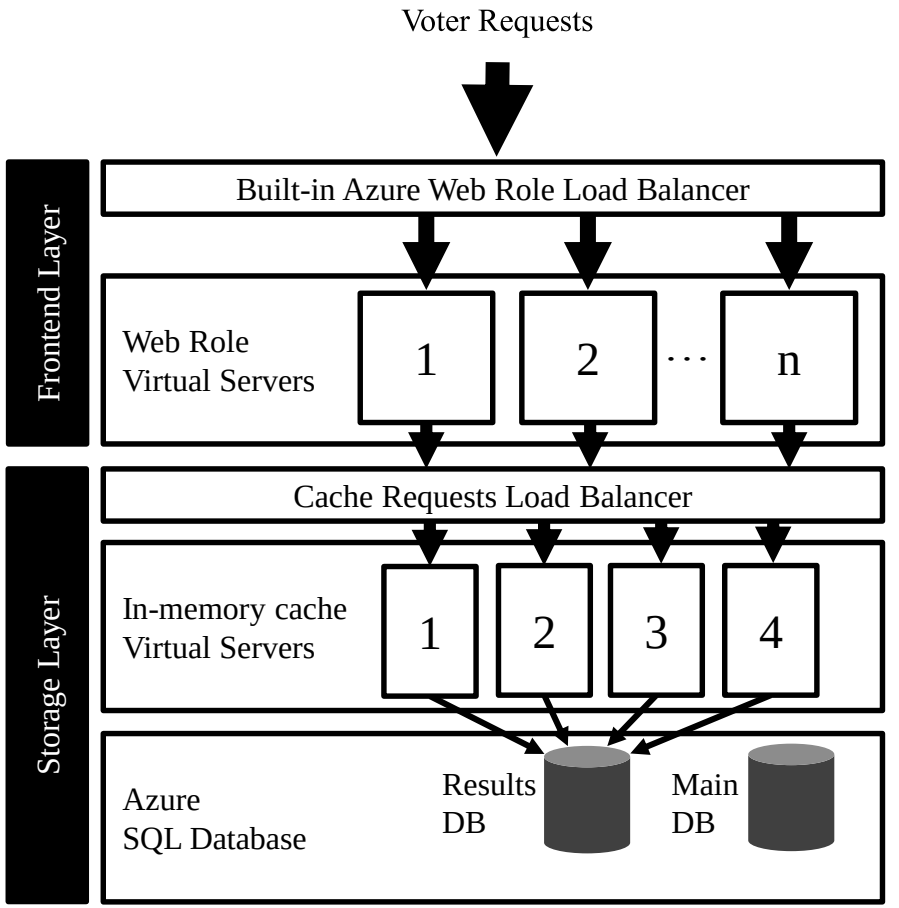
Expected Load				
Scenarios	Expected Page Views	Time Window (hrs)	Page View/sec	10X/pvs DB Calls/sec
Average	10,000,000	4	694	6,944
Peak Hour	6,000,000	1	1,667	16,667

- Problémy:
  - Azure SQL škáluje přibližně do 1000 požadavků za vteřinu

Úprava SW architektury

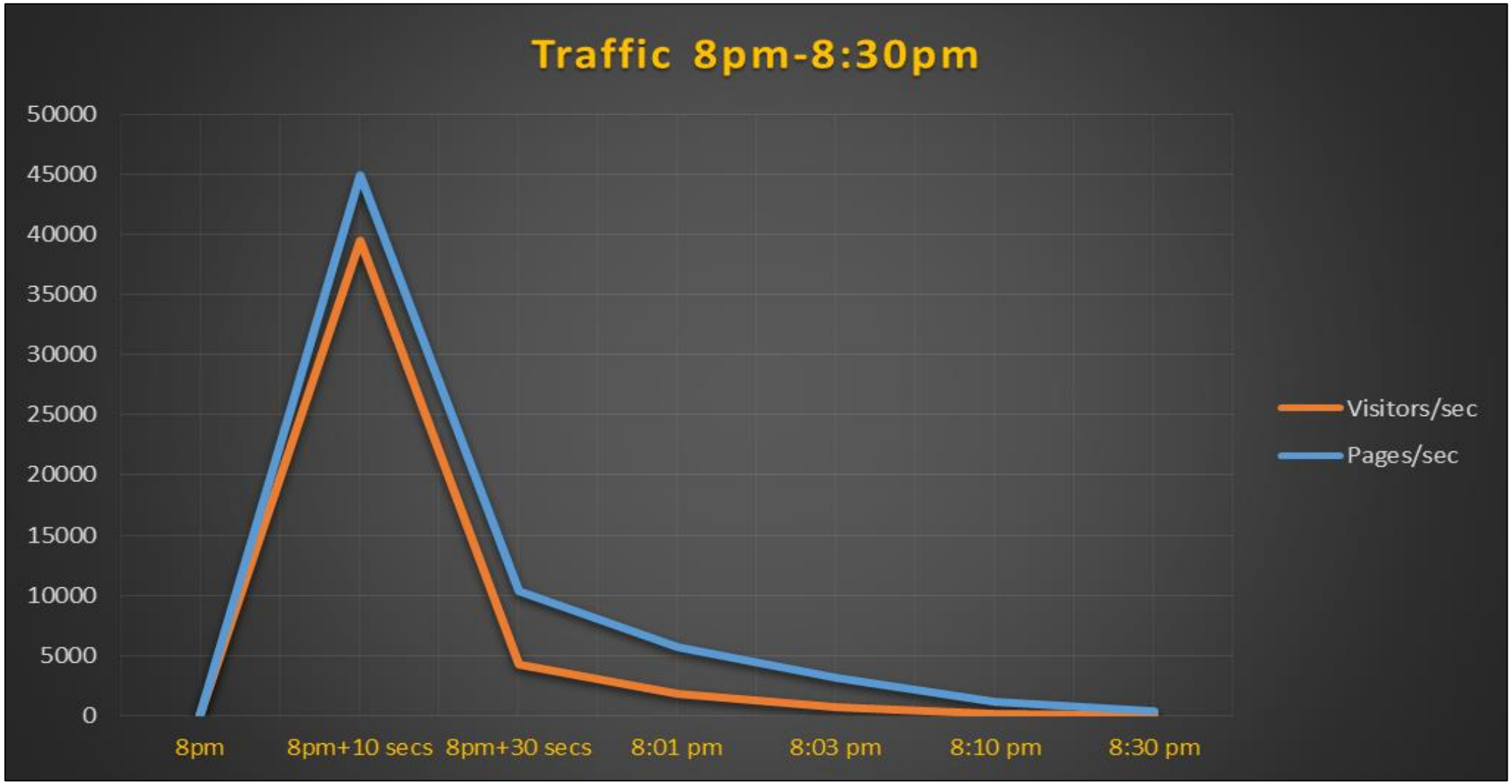
1. Přidání vrstvy **worker serverů** s in-memory cache

- **Řádové navýšení propustnosti úložiště**



Rest of the architecture remains the same

Jak vypadala zátěž



Alokovaná kapacita

- S využitím databáze

Time	Actual Page Views	Time Window (sec)	Page View/sec	Calls/sec	Difference Calls/sec	Requests served
8pm+10 secs	448932	10	44893	448932	-447932	0,22%
8pm+30 secs	206925	20	10346	103463	-102463	0,97%
8:01 odp.	171231	30	5708	57077	-56077	1,75%
8:03 odp.	37835	120	3153	31529	-30529	3,17%
8:10 odp.	494423	420	1177	11772	-10772	8,49%
8:30 odp.	416379	1200	347	3470	-2470	28,82%

- S využitím cache

Time	Actual Page Views	Time Window (sec)	Page View/sec	Calls/sec	Difference Calls/sec	Requests served
8pm+10 secs	448932	10	44893	448932	-288932	35,64%
8pm+30 secs	206925	20	10346	103463	56537	100,00%
8:01 odp.	171231	30	5708	57077	102923	100,00%
8:03 odp.	37835	120	3153	31529	128471	100,00%
8:10 odp.	494423	420	1177	11772	148228	100,00%
8:30 odp.	416379	1200	347	3470	156530	100,00%

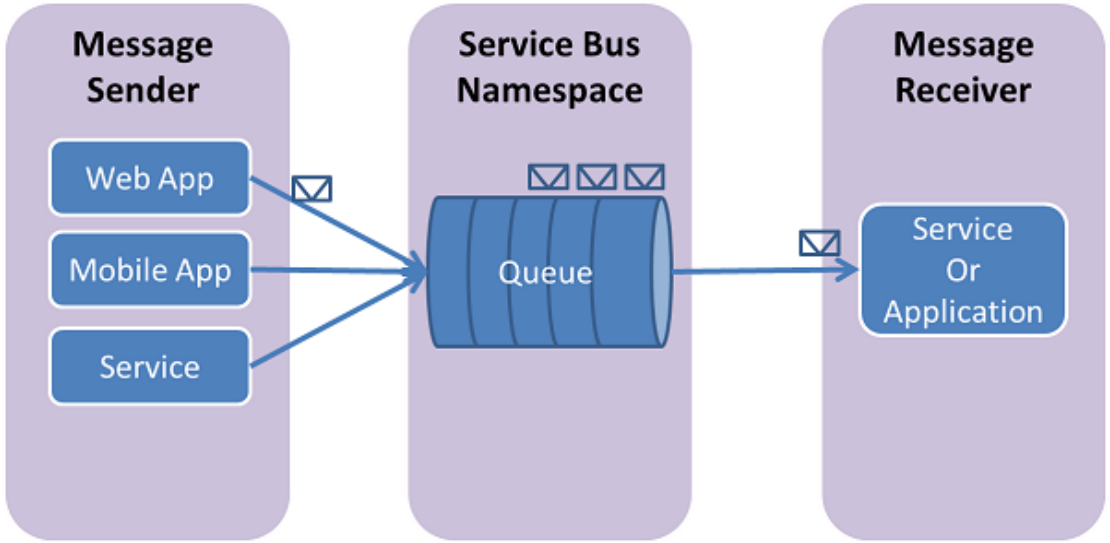
Case study 2: Nákup na Amazonu

- Víte, jak probíhá zpracování stránky při prohlížení produktu na e-shopu Amazon?

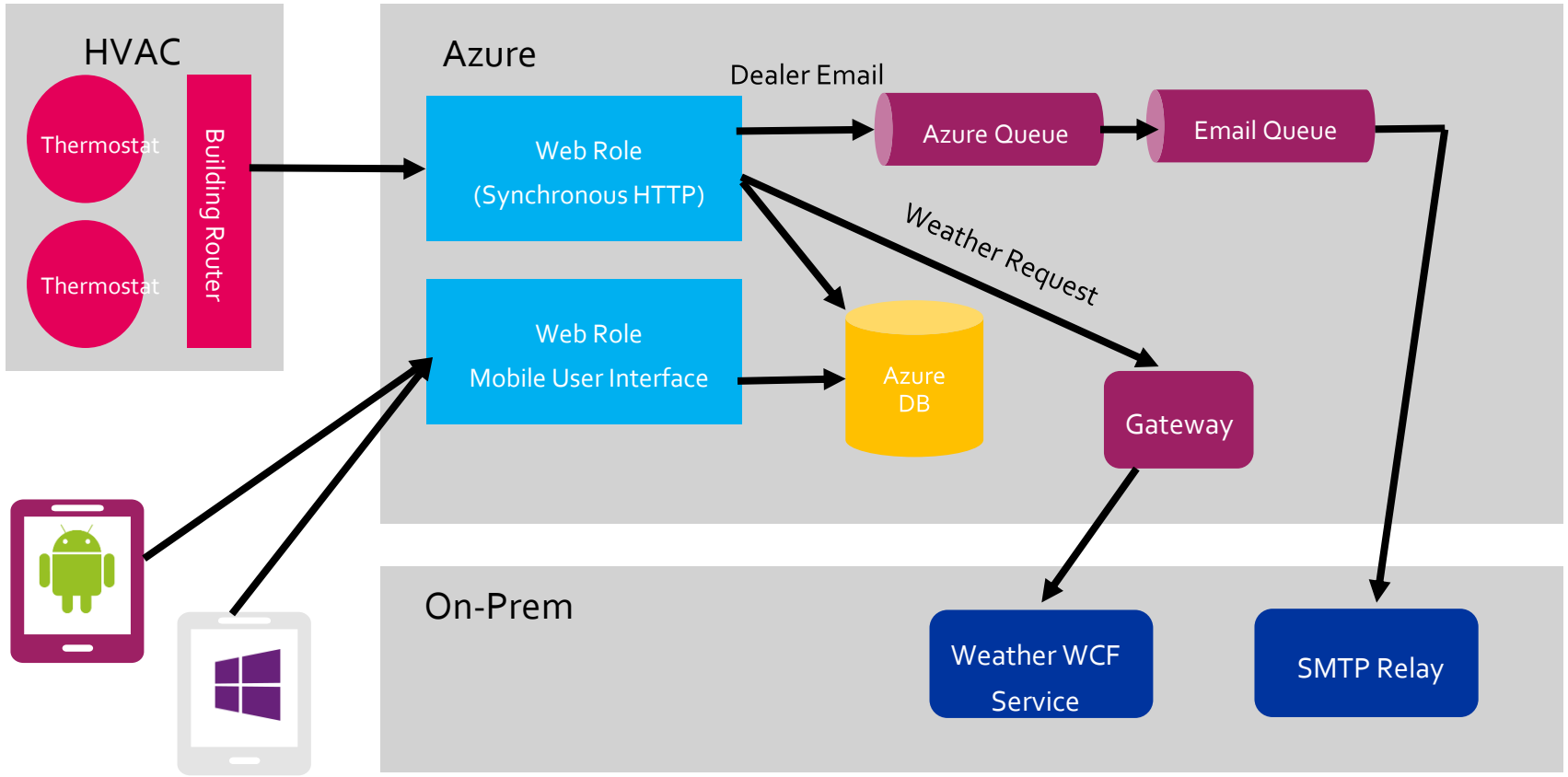
- Stránka včetně doporučených produktů pro uživatele je uložená v úložišti S3 a je pouze načtena bez dalšího zpracování

Asynchronní závislosti

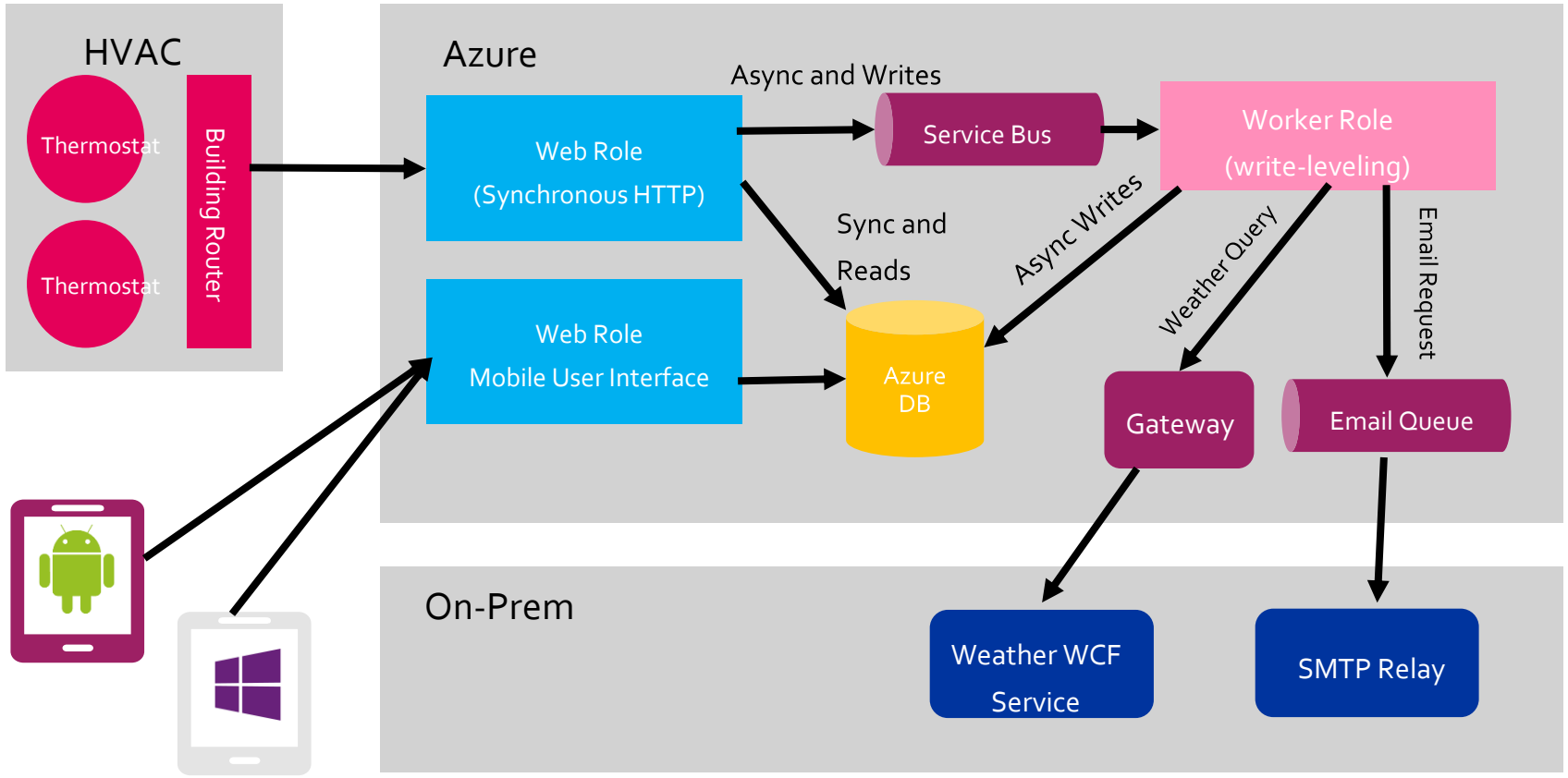
- Výpočetní operace se provádějí asynchronně přes frontu



Case study 3: Chytré termostaty



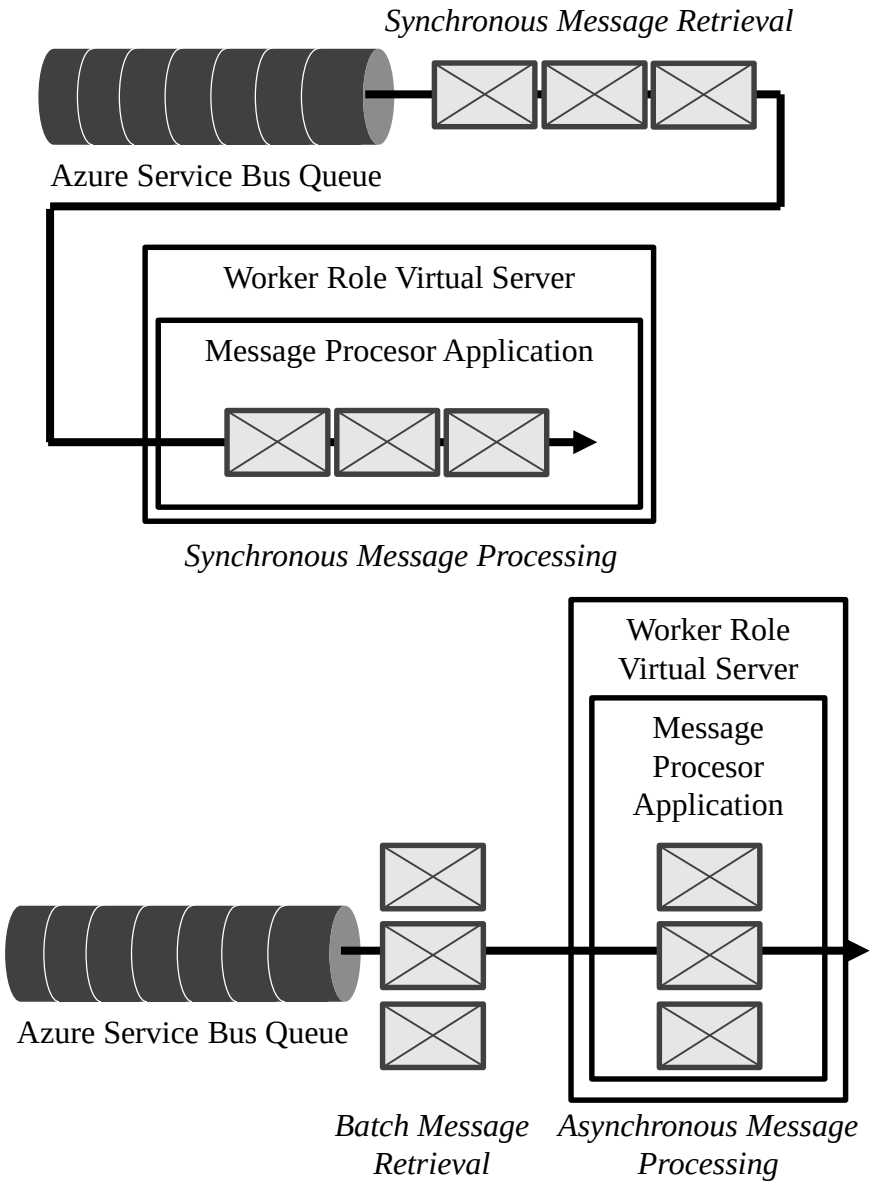
Zanesení asynchronních závislostí



Case study 3: Závěr

- Iniciální testování selhalo už při 35 000 připojených termostatech
- Cíl byl 100 000 (150 000) termostatů

- Hlavní problémy:
  - Synchronní HTTP handler
  - Aktualizace databáze po řádcích (nahrazeno dávkami)
  - Úprava výkonu databáze
  - Navýšení škálovatelnosti fronty jejím partitioningem



Osnova

1. Metriky kvality
2. Case-studies
3. **Taktiky aplikovatelné pro návrh SW architektury cloudové služby**

Taktiky aplikovatelné pro návrh SW architektury cloudové služby

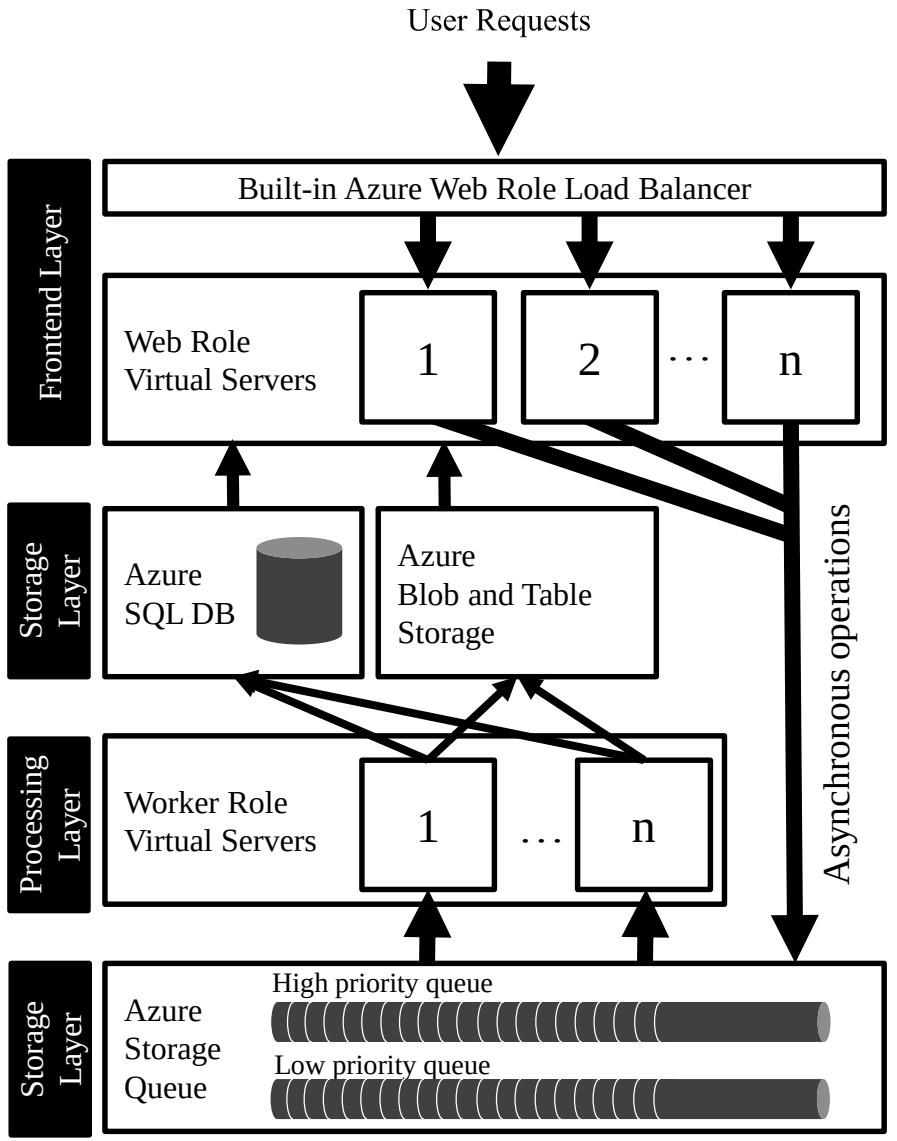
1. Vícevrstvé úložiště
2. Asynchronní závislosti
3. Předpočítání dat

Vícevrstvé úložiště

- Efektivní kombinace výhod různých úložišť:
- **Relační databáze**
  + Transakční zpracování, vynucení integrity dat
  - Omezená škálovatelnost
- **NoSQL databáze** (Azure Table Storage)
  + Dobrá škálovatelnost
  - Složitý návrh klíčů pro efektivní dotazování
- **In-memory cache** (Redis Cache)
  + Extrémní škálovatelnost, rychlá odezva
  - Key/value store, drahé

Asynchronní závislosti

- Požadavky nejsou zpracovány synchronně webserverem
- Jsou uloženy do škálovatelné fronty, postupně zpracovány workery
- Počet workerů je variabilní
- Výsledky uloženy do škálovatelného úložiště
- Problém: Notifikace uživatele o změnách – SignalR (Web Socket)



Předpočítání dat

- Rozšíření varianty vícevrstvého úložiště o předpočítání dat při nevytížených výpočetních zdrojích
- Následné načtení dat probíhá s minimálním vytížením výpočetních zdrojů

- Pro efektivní implementaci vyžaduje aplikaci taktiky asynchronní závislosti

Osnova

1. Metriky kvality
2. Case-studies
3. Taktiky aplikovatelné pro návrh SW architektury cloudové služby