Sofwarové systémy- otázka č. 9

Webové služby a service-oriented architektury, asynchronní architektury komunikace, producer-consumer model, aktivní objekty a agentní systémy. (A4B77ASS)

June 5, 2012

1 Webové služby

Software navržen tak aby podporoval výpočty mezi stroji za použití sítě. Používá WSDL (Web Service Definition Language) pro specifikaci rozhraní. Komunikuje s ostatními službami za pomoci zpráv popsaných formátem SOAP. Typické použití protokolu HTTP za použití XML serializace dat. Použití UDDI (Universal Description, Discovery and Integration) pro nalezení služeb.

Webové služby typy

RESTfull Web Services

- Hlavním cílem je manipulovat s XML reprezentací webových zdrojů
- Jednotný soubor operací nezávyslý na žadateli

"Big" Web Services

- Výstava libovolného souboru operací
- Může být stavový

Realné příklady

Google analytics, blogger, books, calendar, custom search, latitude, maps, translate API, calendar...

E-bay API pro obchochodování

Amazon reklamní API

Busines-to-Busines nejvýznamější prostor pro užívání webových služeb a service-oriented architekture

Vývoj webovýh služeb

Client-Server Jeden centrální server, který obdržuje požadavky od klientů. Například web.

RPC/DCE Framework pro softwarový vývoj, uveden v letech 1990. Microsoft vytvořil svou vlastní alternativu MSRPC. První distribuované objektové systémy byly založeny na RPC/DCE (CORBA, Microsoft DCOM, RMI). Beží taktéž na client-server architektuře (klient pošlo požadavek obdrží výsledek)

XML-RPC objeveno v letech 1990, podpora pro základní datové typy, zprávava byla kódována v XML.

2 Service-oriented architektury (SOA)

Jedná se o soubor zásad a metodiku pro návrh a vývoj softwaru v podobě spolupracujících služeb. Tyto služby jsou přesně definované obchodní funkce, které jsou postaveny jako softwarové komponenty. Mohou být opakovaně použity pro různé účely. SOA principy platí při fázi návrhu a integraci.

Služby jsou strukturovány jako třídy (kombinace informací, zakrývají vnitřní funkčnost, poskytují jednoduché rozhraní pro zbytek aplikace). Služby mohou tvořit hierarchii.

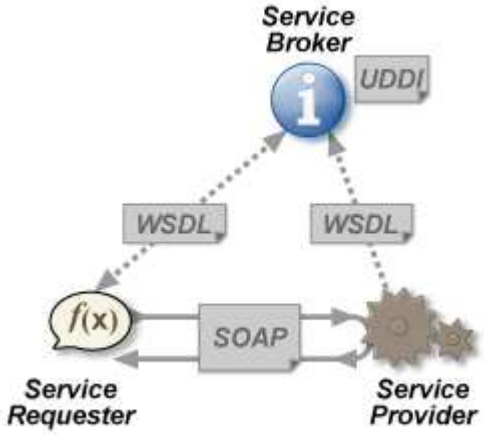
- Poskytuje sadu služeb, která může být použita v rámci více obchodních záměrů.
- Definuje, jak integrovat nesourodé aplikace distribuované na vice systémech
- Definuje rozhraní z pohledu funkcí a protokolů
- Vyžaduje volné spojení se systémem a dalšími technologiemi, které jsou základem aplikace
- Separuje funkce do různých jednotek, nebo služeb, které poskytuje vývojářům po síti

Proč SOA

- Znovupoužití
- Zjednodušené používání starších aplikací
- Platformová nezávyslost
- Vysoká škálovatelnost

Proč ne SOA

- Pokud je důležitý skutečný čas výpočtu


Service
Broker
UDDI
WSDL
WSDL
f(x)
SOAP
Service
Requester
Service
Provider

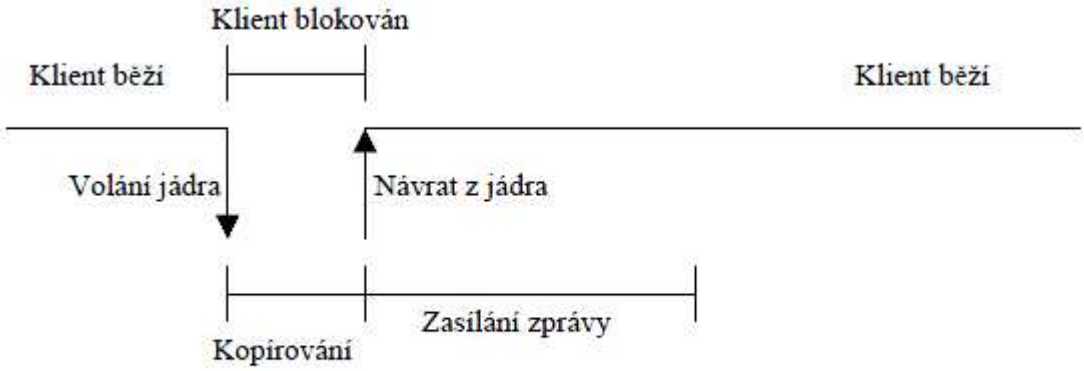
3 Asynchronní architektury komunikace

Asynchorní primitiva:

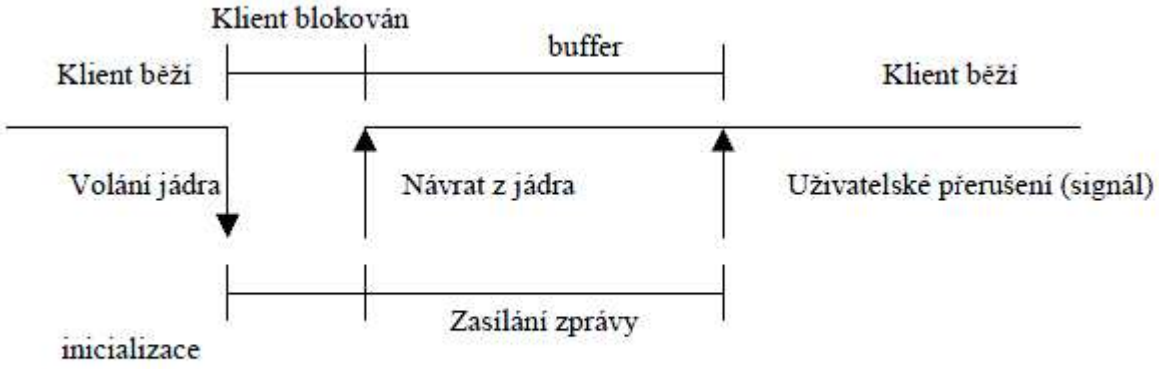
send vrátí kontrolu procesu ihned bez čekání na odeslání zprávy. Výhodou tohoto schématu je vyšší stupeň paralelismu. Proces může pokračovat ve svých výpočtech i po dobu přenosu zprávy. Během přenosu zprávy však proces nesmí používat buffer, ze kterého je zpráva odesílána. To lze vyřešit dvěma způsoby:

1. Jádro systému si okopíruje zprávu do svých vnitřních bufferů, čímž umožní procesu další volný běh.
2. Po odeslání zprávy je proces přerušen.

receive pouze říká jádru, kam má umístit došlou zprávu. I zde můžeme využít např. tzv. primitiva wait, které pozastaví proces do té doby, než zpráva dorazí. Druhou možností je zavedení primitiva test, které otestuje, zda je nějaká zpráva připravena nebo conditional_retrieve, které buď vybere došlou zprávu nebo se vrátí s informací, že žádná zpráva není k dispozici. Poslední možností je zavedení uživatelského přerušení, které bude informovat o došlé zprávě.



Asynchronní komunikace s kopírováním



Asynchronní komunikace s přerušením

4 producer-consumer problém

Jedná se o typický problém multiprocesové synchronizace. Problém popisuje 2 procesy, producentů a konzumentů, kteří používají jeden buffer používaný jako frontu s pevnou velikostí. Producent generuje data, ukládá je do bufferu a začíná je znovu generovat. Ve stejný čas konzument používá data z bufferu. Problém je, že producent nemůže ukládat data do bufferu, pokud je plný a konzument nemůže brát data, pokud je buffer prázdný.

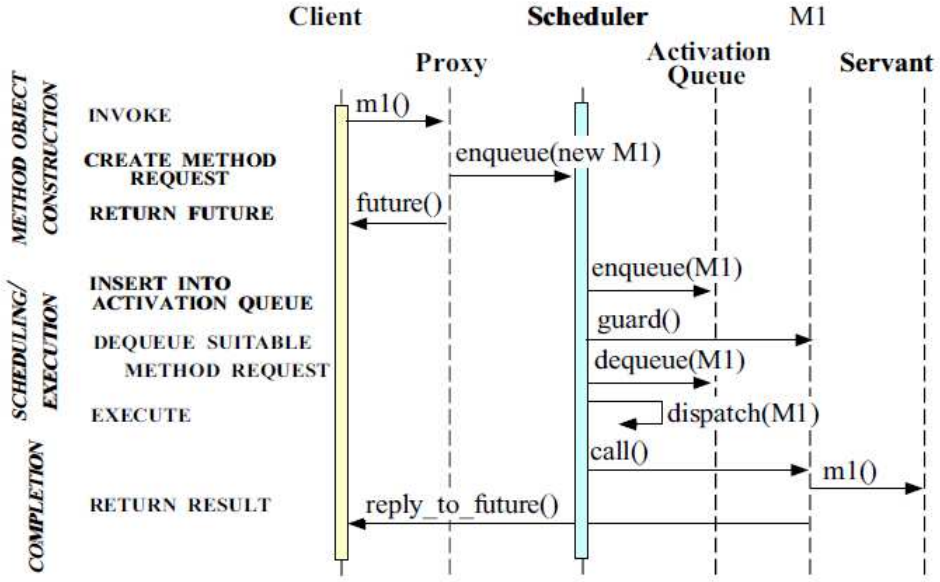
Řešením tohoto problému je, uspat producenta, pokud je zásobník plný a probudit ho, když se uvolní místo. Zároveň pak uspat konzumenta pokud je zásobník prázdný a probudit ho, pokud je v zásobníku něco vloženo.

- Řešení přes standartní sleep, wakeup, kdy se při každé iteraci kontroluje stav zásobníku. Může docházet k deadlocku
- Řešení přes semafory. Procesy si dávají navzájem přednost podle toho v jakém stavu je semafor.
- Řešení přes monitory. Synchornizaci provádí akce add a remove nad zásobníkem. Pokud je zásobník prázdný zamknou konzumenta...

5 Aktivní objekty

Jedná se o návrhový vzor, který odděluje metody provádění výpočtu a volání. Zlepšuje souběžnost a snižuje problémy se synchronizací. Na těchto objektech bývají velmi často založeny Agentní a Multiagentní systémy.

Princip activního objektu je založen na tom, že je vyvolán asynchroní výpočet, kterému jsou předány vstupní parametry a návratová funkce. Objekt který vyvolá asynchroní akci, zahodí na výpočet referenci, a pracuje dál bez ohledu na externím výpočtu (ne tak so slova, pokud je výpočet nezbytný, čeká na něj). Externí výpočet se sám ozve na referenci kterou v sobě uchovává.



6 Agentní systémy

Speciální podmnožina umělých inteligencí. Zkoumá koncept automatického rozhodování, komunikace a koordinace, distribuovaného plánování a učení. Herni aspekty jako chování soutěžícího, nebo logickou formalizaci vyšších struktur na úrovni znalostí.

Agent je zapouzdřený výpočetní systém, který se nachází v nějákém prostředí, a je schopen pružného, samostatného chování za účelem splnění svého cíle. Agent může existovat samostatně, ale často je součástí multi-agentního systému.

Technologie Agentů poskytuje sadu nástrojů, algoritmů a metod pro vývoj a distribuovaných a asynchronních inteligentních softwarových aplikací.

Klíčové vlastnsoti:

- Samostatnost - agent rozhoduje sám za sebe a nemá žádný dozor z venčí
- Reaktivita - agent je schopen rychle reagovat na události v prostředí
- Schopnost zachovat dlouhodobé plány a zvažování dalších kroků k dosáhnutí plánu.
- Sociální schopnosti - je schopen komunikovat a spolupracovat.

Modely užití agenta:

- agent jako metafora - pomáhá vývojářum vývoj ohledně samostatnosti v komunikaci
- zdoj technologii
- simulační - poskytuje simulaci problému realného světa

Úrovně návrhu agenta:

- Organizační úroveň
- Interakční úroveň - komunikace mezi agenty
- Agentní úroveň - týká se samotného agenta (učení...)

Užití agentů:

- Zeměplošné rozložení
- Konkurenční domény

- Časově kritické odevy a vysoká odolnost
- Simulace a modelování scénářů
- Open system scénáře
- Complex system scénáře
- Samostatnostně orientované aspekty