VÍCEÚROVŇOVÉ DATABÁZOVÉ KLASTRY V PROSTŘEDÍ ROZSÁHLÝCH WEBOVÝCH INFORMAČNÍCH SYSTÉMŮ.

Miroslav Křipač
Michal Brandejs

Masarykova univerzita v Brně, Botanická 68a, 602 00 Brno, ČR,
E-mail: kripac@fi.muni.cz, brandejs@fi.muni.cz

Abstrakt
Databázové klastry představují v poslední době významnou roli při implementaci rozsáhlých informačních systémů. Kromě tradičních vlastností jako je zajištění provozu služby v případě výpadku mohou umožnit také efektivnější rozložení a nárůst výpočetního výkonu, neboť jedna databáze může být obsluhována více fyzicky nezávislými servery. Klastry se tak stávají základem pro nasazení modelu GRIDu i v oblasti podnikových informačních systémů. Přesto každá aplikace není pro nasazení v tomto prostředí vhodná, protože škálovatelnost jejích požadavků na databázové úrovni nemusí být při jejich zvyšujícím se množství požadavků dostatečná. Tento článek má za cíl představit alternativní dosažení zmíněných vlastností, které je založeno na kombinaci různých klastrových řešení v rámci jednoho databázového systému na více úrovních. Vzniklá infrastruktura je navíc navržena s ohledem na nasazení pokročilých metod detekce a zabránění přetížení systému, která mohou vznikat zejména v souvislosti s využitím ve striktně webových informačních systémech. Praktická část příspěvku bude věnována konkrétní implementaci takto navrženého modelu a výběru vhodných technologií, které jsou pro tento účel dostupné. Produkční nasazení pak bude demonstrováno na příkladu Informačního systému Masarykovy univerzity v Brně, který model víceúrovňových databázových klastrů úspěšně využívá při integraci velkého množství služeb v rámci rozsáhlé heterogenní organizace s desítkami tisíc denně přistupujících uživatelů.

1. ÚVOD

S postupným nasazováním informačních systémů v masivním prostředí pro mnoho současně přistupujících uživatelů nabývá na důležitosti vhodné navržení architektury, které bude i při enormním nárůstu požadavků zajišťovat dostatečnou odezvu a dostupnost služeb poskytovaných systémem. V souvislosti s návrhem systému se o architektuře obvykle uvažuje spíše z návrhového a následně implementačního hlediska, přičemž se často, a nutno říci že také správně, klade důraz zejména na výběr takových prostředků, které vedou k efektivní realizaci všech úrovní aplikace, její údržby a rozšiřitelnosti. Objevují se tak například postupy, které vedou ke striktnímu oddělení aplikační a prezentační logiky na jedné straně, a přístupu k datům na straně druhé tak, aby programátor aplikací měl možnost pracovat s daty bez ohledu na jejich skutečnou různorodou reprezentaci [1]. Aplikace se tak mohou stát snadněji přenositelné a přizpůsobitelné novým prostředím (datovým zdrojům). Do těchto úvah můžeme zařadit také otázku vhodnosti použití konkrétního paradigmatu návrhu aplikací a datového modelu, otázku objektově orientovaného přístupu apod.[2].
Odhlédněme však nyní od těchto problémů spíše ke konkrétní realizaci daného, již navrženého, systému a přihlédněme zejména k jeho konkrétnímu provozu v reálném prostředí. Pokusíme se proto bez ohledu na to, jaký byl zvolen přístup k analýze funkčního a datového modelu, určit některé hlavní a přirozené vlastnosti, které by měly být pro libovolný systém závazné, a navrhnout takové prostředí, které by je mohlo realizovat. Úvahy, kterými se budeme zabývat, však obvykle získávají na významu teprve ve chvíli, kdy dojde k velkému nárůstu požadavků na systém, proto nemusí být pro většinu realizovaných projektů

relevantní. Přesto by mohlo být zajímavé zajistit počátečním návrhem architektury vhodnou platformu pro předpokládaný růst objemu zpracovávaných operací v systému. Dobře navržená architektura může být proto pro realizaci téže aplikace a stejného datového modelu efektivní jak v malém měřítku pro řádově stovky uživatelů, tak pro desítky tisíc aktivních uživatelů, pokud se při realizaci použijí takové adekvátní prostředky, aby nedošlo k neefektivnímu předimenzování některé části nebo i systému jako celku.

2. ARCHITEKTURA SYSTÉMŮ

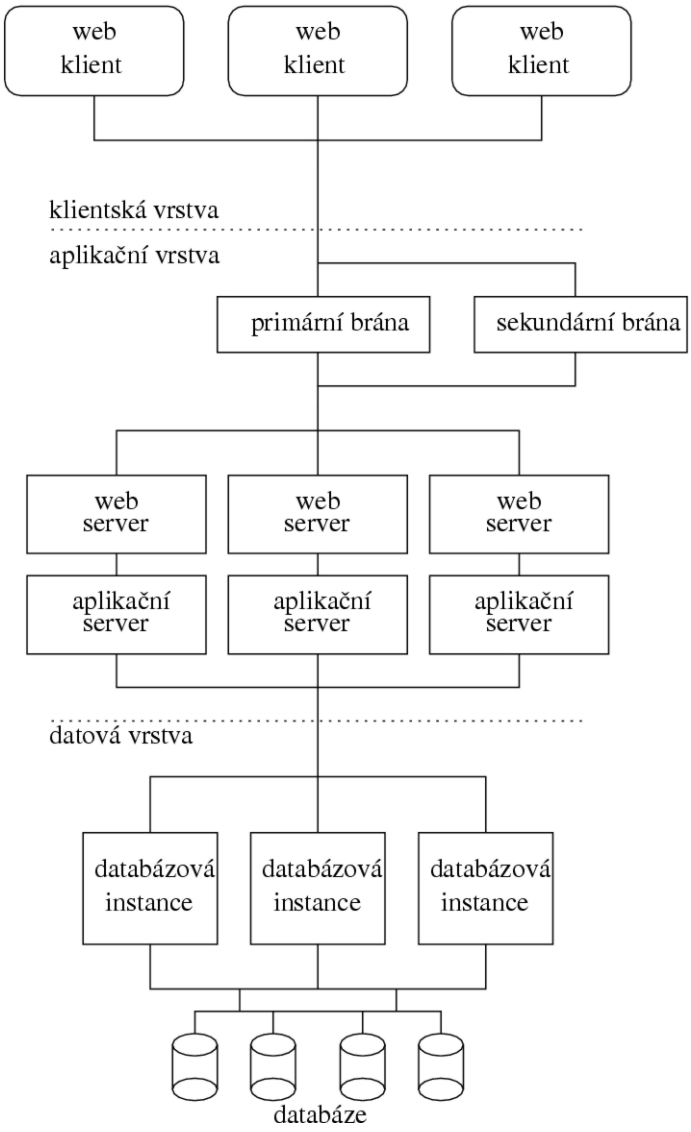
Jedním z hlavních požadavků současných informačních systémů je jejich snadná dostupnost pokud možno pro všechny potenciální uživatele, ať už se jedná o samotné členy dané organizace, klienty, partnery apod. Zejména přístup z různých druhů vysoce heterogenních prostředí, byť jen k určité části služeb a informací spravovaných systémem, se dnes již stává samozřejmostí. Předpokládejme tedy, že klasický model klient-server, kdy uživatel udržuje veškerou aplikační logiku pod svou správou na klientské straně a centrální část zajišťuje pouze přístup k datům, je nanejvýš vhodné nahradit nyní běžně používaným třívrstvým modelem. Ačkoliv se to může zdát zřejmé, v praxi se stále setkáváme se systémy, které na klasickém modelu trvají. Výhoda není pouze ve snadném přístupu k systému, ale zejména ve snadné distribuci a správě aplikací, které jsou udržovány centrálně. I přes jistě náročnější implementaci, která nemůže snadno využít všech možností hostující platformy pro klientskou část, se třívrstvá architektura založená na přístupu prostřednictvím webového prohlížeče jeví pro úspěšné nasazení v masivním prostředí jako nezbytná. V dalších myšlenkách se proto budeme věnovat výhradně systémům založeným striktně na tomto webovém přístupu.

První částí systému je tedy webový prohlížeč, který je jako jediná část instalován přímo na straně klientského počítače nebo třeba mobilního telefonu. Uživatel tak může se systémem začít pracovat, aniž by si jeho počítač vyžádal zásah správce či víceméně zkušenějšího uživatele. Zbytek systému je plně pod kontrolou na straně jeho provozovatele. Tato část obvykle obsahuje prezentační, aplikační a datovou vrstvu, přičemž z hlediska konkrétní implementace lze prezentaci (tj. realizaci konkrétní komunikace s webovým prohlížečem a generování dat na výstupu ve vhodném formátu pro prezentaci) a aplikaci považovat za jeden celek, byť je logicky nutné ji rozdělit do různých procesů z důvodu zmiňovaného návrhu a údržby aplikací. Poslední datová vrstva realizuje efektivní uložení dat evidovaných v rámci systému, a pokud možno rychlý přístup k nim ze všech složek aplikační úrovně.

Důležitou vlastností takto navržené architektury je přesné rozlišení jednotlivých vrstev v práci s daty. Aplikační vrstva předpokládá, že data jsou uložena v databázovém subsystému, a proto není vhodné na její úrovni implementovat žádný mechanismus pro souběžný přístup k datům. Pokud data striktně poskytuje a udržuje datová vrstva, může se zátěž systému rozdělit rovnoměrně mezi fyzicky oddělené servery, takže výpočetní výkon je rozložen. Implementace masivního současného přístupu k vysoce strukturovaným datům je přirozenou otázkou datové vrstvy a databázové systémy toto obvykle poskytují na nejvyšší možné úrovni. Proto je vhodné na prostřední aplikační úrovni uchovávat pouze ta data, která jsou víceméně permanentní a jejichž změny si nevyžádají žádnou významnou režii. Tím lze aplikační procesy jednoduše distribuovat na fyzicky navzájem nezávislé stroje, neboť synchronizace současného přístupu k datům bude realizována výhradně v další vrstvě.

Vraťme se však nyní k otázce hlavních vlastností, které musí vhodná architektura pro masivní nasazení splňovat. Zaměříme se na dvě z nich: vysoká dostupnost systému a škálovatelnost ve smyslu snadného rozložení zátěže a navýšení výkonu. Dostupnost lze chápat velmi široce. Zejména v případě systému otevřeného pro přístup nepřetržitě všem uživatelům musí být systémová infrastruktura odolná nejen proti výpadkům a chybám, ale také proti odstávkám z důvodu plánovaných údržbových zásahů nemluvě o odstávkách aplikací kvůli pravidelným hromadným operacím (dříve bylo nezbytné zamezit přístupu uživatelů vždy v době výpočtu

různých uzávěrek či během pravidelného zálohování). Přístup dvacet čtyři hodin denně není v současné době jen reklamním lákadlem firem nabízejících příslušné technologie - v reálném provozu se s tímto požadavkem setkáváme již zcela běžně.



Obr. 1 Realizace třívrstvé architektury v distribuovaném prostředí

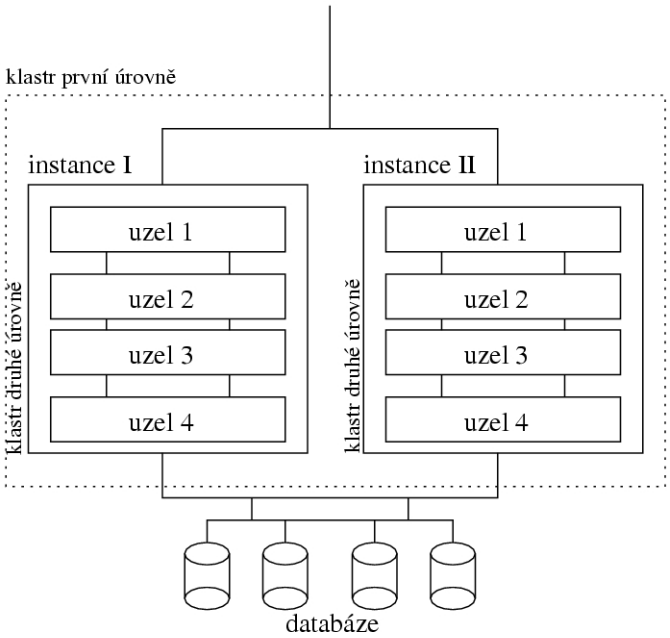
Z technického pohledu lze vysokou dostupnost zajisti pouze násobením jednotlivých zdrojů, které se v systému využívají. V případě výpadku jedné části pak převezme provoz odpovídající druhá složka. Toto násobení zdrojů je vhodné zajistit nad fyzicky nezávislými částmi formou distribuovaného zpracování požadavku na všech úrovních zpracování od síťové brány systému až po znásobování diskového subsystému i jeho částí na datové úrovni. Fyzické oddělení částí sloužících ke stejnému účelu zvyšuje míru izolace případného problému (softwarové, ale i hardwarové chyby) a umožňuje provádět údržbu postupně za provozu systému (v drtivé většině zásahů).
Distribuované zpracování může na jednotlivých vrstvách systému zajistit kromě vyšší míry dostupnosti také velmi efektivní rozšiřitelnost systémů a rozložení zátěže v případě nárůstu požadavků. Zejména v případě, kdy je k datům striktně přistupováno přes datovou vrstvu, může být navýšení výkonu na aplikační vrstvě velmi jednoduché, neboť přístup k datům je

v tomto případě transparentní a vnitřní mechanismy pro distribuci vyrovnávacích pamětí neměnících se dat lze obvykle realizovat bez navýšení režie i při vyšší distribuovanosti. Navýšení požadovaného výkonu tak znamená obvykle jen přidání dalšího aplikačního serveru namísto přidání jednotlivých zdrojů (procesoru, paměti) ve víceprocesorových prostředích. Rovněž databázová vrstva v některých případech umožňuje nasadit takto distribuovaný způsob.
Z výše uvedeného plyne, že třívrstvá architektura webových informačních systémů se může postupně přesunout z realizace na výkonných víceprocesorových platformách do modelu vzájemného propojení jednoduchých komoditních serverů na obou úrovních, které dohromady poskytnou stejný nebo i vyšší výkon s nižšími náklady. Podobně jako v prostředích pro náročné vědecko-technické výpočty se tak začíná i v oblasti podnikových informačních systémů prosazovat model tzv. GRIDů, tedy sítí vzájemně nezávislých počítačů, které zajišťují jednu službu. Naše zkušenosti a měření z různých forem testovacích provozů však ukazují, že koncept databázových klastrů není vhodný pro každou aplikaci, a to ne z důvodu nekompatibility řešení jako takového, ale výhradně díky nedostatečnému výkonu při nasazení v běžném provozu a škálovatelnosti datové vrstvy. Takto zmíněný návrh architektury, který je jinak vhodný pro řadu prostředí, je proto v některých podmínkách nutné přepracovat. Naše cesta vedla k vytvoření nového modelu takzvaných víceúrovňových databázových klastrů, který se ukazuje být vhodnou alternativou pro běžné databázové klastry, jak je známe z konkrétních implementací.

3. VÍCEÚROVŇOVÉ DATABÁZOVÉ KLASTRY

Cílem návrhu modelu víceúrovňových klastrů bylo zachování obou zmíněných vlastností systému - vysoké dostupnosti a snadné škálovatelnosti. Ukazuje se však, že na úrovni datové vrstvy třívrstvé architektury je nezbytné dosáhnout těchto vlastností odděleně. Základem pro dosažení vysoké dostupnosti je opět rozdělení databázové instance (pod pojmem databázová instance rozumíme množinu procesů v rámci jednoho běhu operačního systému, které slouží ke správě jedné databáze a zajišťují přístup k datům uloženým v databázi) na více vzájemně propojených fyzicky nezávislých serverů tak, jak to realizují databázové klastry. Přirozenou vlastností takového rozdělení je možnost přistupovat (číst i zapisovat) ke všem datům pomocí libovolné instance transparentně. Takto vzniklý klastr pak vytváří příhodné prostředí pro zachování služby v případě výpadku (ať už plánovaného nebo mimořádného z důvodu chyby). Nastane-li taková situace, je možné aplikace směřovat vždy pouze na ty instance, které nebyly výpadkem dotčeny, přičemž ovlivnění ostatních instancí chybou jedné z nich je minimalizováno. Oproti typickým databázovým klastrům však v tomto modelu nedochází k přímé distribuci zátěže mezi všechny aktivní instance databáze, ale zátěž je řízeně směrována pouze na jednu (primární) instanci. Důvodem je skutečnost, že ne každá aplikace je schopná vyvolat na softwarové úrovni databázového systému takovou škálovatelnost, aby režie na provoz neovlivnila svoji vlastní výkonnost. Toto schéma se jakoby vrací zpět od skutečně distribuovaných databázových systémů k různým alternativám, kdy klastr tvoří pouze dvě instance - aktivní a záložní, přičemž záložní je připravena převzít provoz, ale není ve skutečnosti využívána pro žádné požadavky. To je jistě výrazně jednodušší na implementaci, nicméně neefektivní z hlediska využití všech dostupných zdrojů. Model víceúrovňových klastrů realizuje jakýsi kompromis, kdy hlavní provoz je sice z důvodu výkonu směřován na jednu instanci, ale ostatní instance také mohou sloužit pro zpracovávání přesně specifikovaného typu požadavků. Například je možné na tyto instance směřovat dávkové úlohy přímo nad on-line databází či zkušební provoz. Alternativně lze sekundární instance využít při konstrukci sofistikovaných metod zabraňujících přetížení systému.
I přes zmiňovaný kompromis však takto nastavený klastr pozbývá druhé zásadní výhody, snadné rozšiřitelnosti systému, neboť hlavní provoz je vždy směřován na jednu instanci.

Přesto se ukazuje, že jednu databázovou instanci je možné realizovat opět pomocí klastru více uzlů jisté druhé úrovně. Takovýto klastr musí zajistit, že fyzicky nezávislé avšak propojené uzly tvoří dohromady jedno prostředí pro běh aplikace. Tak jako na první úrovni tvořily jednotlivé nezávislé instance platformu pro obsluhu jedné databáze, klastr na druhé úrovni tvoří platformu pro běh jedné instance. Z definice instance plyne, že takový klastr musí být vůči procesům v operačním systému transparentní, tedy je nutné jej realizovat blíže k hardwarové úrovni systému. Naopak ale takovýto klastr nemusí splňovat druhou podmínku, nezávislost uzlů při výpadku, neboť dostupnost systému jako celku je realizovaná na první klastrové úrovni, tedy mimo celý klastr druhé úrovně. To umožňuje snadnější implementaci vysoce škálovatelného klastru.



Obr. 2 Schéma modelu víceúrovňového klastru

Každá úroveň takto implementovaného databázového systému splňuje právě jednu z navzájem protichůdných vlastností, které jsme zmínili pro úspěšný chod současného systému. Díky tomu, že jednotlivá vrstva je optimalizována pouze na jednu vlastnost, můžeme ji realizovat mnohem snadněji, než pokud by se snažila o implementaci obou. Výsledkem je vysoce dostupný databázový systém složený z více nezávislých databázových instancí, který je snadno škálovatelný pouhým přidáním dalších uzlů v rámci každé instance. Jediným omezením pro takto navržený model distribuce zpracování databázového požadavku je posun od vysloveně komoditních systémů ke klastrům, které jsou implementovány na hardwarové úrovni. Přesto se ukazuje nasazení takto navrženého systému efektivnější a přijatelnější, než využití klasických multiprocesorových platforem, z nichž navíc pouze malá část ve skutečnosti implementuje dostupnost na úrovni fyzické nezávislosti. Také rozšiřitelnost běžných multiprocesorových platforem je oproti technice postupného přidávání nezávislých uzlů vždy dána infrastrukturou, kterou má daný systém pro budoucí rozšíření předem připravenou.

4. IMPLEMENTACE

Výhody, které jsme představili v návrhu modelu víceúrovňových klastrů, je však nutné realizovat v konkrétním prostředí. Představme si nyní technologie, které jsou pro takový účel vhodné, případně se dají pro něj upravit. První úroveň našeho databázového klastru je tvořena systémem nezávislých databázových instancí, které obsluhují přístup k jedné databázi. Pro její

realizaci se ukazuje jako vhodné vyjít ze základního konceptu, který přináší produkt Oracle Database již v základní verzi s rozšířením Real Application Clusters[3]. Tato technologie umožňuje spojení dvou nebo více nezávislých serverů jednak mezi sebou na síťové úrovni, jednak s diskovým subsystémem na úrovni datové. Jednotlivé uzly pak mohou přistupovat transparentně ke všem datům v diskovém subsystému a jeden může nahradit druhý v případě výpadku. Původní implementace společnosti Oracle však neumožňuje dostatečně škálovat každou aplikaci, která tímto způsobem k datům přistupuje. Naše řešení proto přináší i s využitím této technologie možnost směrovat zátěž řízeně na jednu primární instanci za současného zachování funkčnosti ostatních instancí a požadovaného škálování výkonu. Tato modifikace se ukazuje jako dostatečná nejen pro třídu aplikací z našeho testovacího prostředí (tedy zejména zmiňované aplikace webového informačního systému). Hlavní výhodou, kterou přináší, je zachování provozu databázové služby v drtivé většině výpadků, které by jinak vedly k havárii celého systému.

Druhá úroveň klastrů předpokládá, že techniky distribuce výpočtu do více uzlů budou transparentní vůči procesům na úrovni operačního systému. Takový klastr je tedy nutné realizovat s významnou podporou na hardwarové úrovni. Pro toto prostředí se ukazuje jako vhodná technologie NUMAlink společnosti Silicon Graphics (SGI), která vychází z dlouholetého konceptu architektury globální sdílené paměti[4]. Původním cílem této technologie je výstavba vysoce škálovatelných výpočetních prostředí s architekturou typu NUMA, která je založena na technice klastrování fyzicky nezávislých uzlů, přičemž každý uzel může volitelně zajišťovat různé funkce v rámci klastru (výpočetní uzly, uzly pro vstupně-výstupní operace a jejich vzájemná kombinace). Ačkoliv je původní určení této technologie zcela odlišné od našeho zaměření na využití v širším kontextu databázových klastrů, ukazuje se, že vhodnou úpravou lze takto dostupné řešení pro naše prostředí využít. Získáme tak platformu pro běh jedné z databázových instancí, která má bezpochyby nejvyšší poměr mezi pořizovacími náklady a výkonem, kterého lze efektivně dosáhnout. Navíc teoretická škálovatelnost takového prostředí v reálném provozu je až 256 procesorů (prakticky byly dosud v prostředí databázových systému nasazeny menší systémy, přičemž naše zkušenost ukazuje téměř lineární škálovatelnost nejméně do šestnácti procesorů v jednom klastru).

5. PŘÍKLAD Z PRODUKČNÍHO PROVOZU

Přestože se nám podařilo pro daný model vybrat vhodné implementační prostředky, může být takové řešení stále spíše teoretickým projektem. Pro nasazení v praxi je jistě nutné je použít v konkrétním provozu, kde se teprve daná metoda skutečně prověří. V kontaktu s inženýry společnosti SGI se nám podařilo sestavit v tomto smyslu unikátní řešení[5], které slouží ke zpracování dat v rámci databázové vrstvy Informačního systému Masarykovy univerzity v Brně. Ukazuje se, že stejné technologie je možné využít také jako platformu pro zajištění infrastruktury doslova všech aplikací, které jsou schopné spolupracovat s databázovým systémem Oracle Database.

Informační systém Masarykovy univerzity je systém s výhradně webovým přístupem pro zpracování kompletní studijní agendy, která se v prostředí velké veřejné vysoké školy objevuje. Systém tak zajišťuje elektronicky vše související se studiem od elektronického podání přihlášky ke studiu, přes registraci kurzů a hodnocení studentů až po automatickou kontrolu individuálního studia, kdy si student sám volí předměty libovolné fakulty a sestavuje rozvrh, a na základě ní následný tisk diplomů pro úspěšné absolventy. Snahou systému je integrovat velice různorodé prostředí (Masarykova univerzita má v současné době devět fyzicky izolovaných fakult s různým zaměřením) a usnadnit všechny administrativní, komunikační, rozhodovací a samotné studijní procesy využitím dostupných informačních technologií. Takováto integrace však v prostředí s více než 30 000 aktivními uživateli, z nichž přibližně polovina využívá systém každý den, vyžaduje na straně systému velmi efektivní

využití zdrojů, neboť systém je postaven na filosofii striktního on-line zpracování všech požadavků (téměř každý požadavek není zpracováván postupně frontou požadavků, ale přímo při předání uživatelem). Systém tak udržuje všechna data (snad s výjimkou některých statistik) stále aktuální, což stále není běžné ani v jiných oblastech využití informačních technologií.
V minulém roce došlo proto uvnitř databázové části k přechodu na metodu víceúrovňových klastrů. Samotný proces realizace nebyl vůbec jednoduchý, protože se při něm vyskytla řada dosud neznámých problémů. Přesto se všechny z nich podařilo vyřešit ať už softwarovou opravou chyby nebo využitím alternativních postupů vedoucích ke stejnému cíli. Výsledkem je stabilní systém odolný proti drtivé většině různých výpadků, který při svém dosud největším zatížení zvládal vyřizovat více než jeden milión klientských požadavků za den. Zajímavá pro nasazení této technologie je rovněž skutečnost, že k masivnímu navýšení kapacit došlo při dodržení velice nízkých pořizovacích nákladů, které nepřesáhly 300Kč na jednoho aktivního uživatele s tím, že předpokládané využití této investice je při stávajícím růstu termínováno nejméně na tři roky. Finanční otázka, která je velmi důležitá nejen při financování z veřejných zdrojů, rovněž přispívá k úspěchu této metody v reálném prostředí.

6. ZÁVĚR A DALŠÍ ROZŠÍŘENÍ

Cílem článku bylo nabídnout zájemcům alternativní možnosti realizace rozsáhlých informačních systémů. Je zřejmé, že dané prostředí nemusí být efektivní pro nasazení v menším prostředí, kdy dostupnost ani rozšiřitelnost není tolik kritická. Ani Informační systém Masarykovy univerzity, který je provozován na několika dalších českých univerzitách a nabízen MU k využití, nevyužívá všechny možnosti víceúrovňových klastrů v provozu řádově menších organizací. Jako důležité se však ukazuje, že původní architektura aplikací a rozhraní jednotlivých komponent v rámci systému je schopná v případě nárůstu požadavků na jednotlivé parametry jednoduše všechny zmiňované výhody využít. Třívrstvá architektura pak dovoluje bez problému provozovat jádro systému (aplikace a data) fyzicky odděleně od hlavního sídla organizace, kde je soustředěno největší množství uživatelů, což podle našich dlouholetých zkušeností výrazně snižuje náklady nejen na lidské zdroje.
Metoda víceúrovňových klastrů však nabízí ještě další dosud plně nevyužité možnosti. V on-line systémech založených na webovém přístupu může snadněji docházet k lokálním (dočasným) přetížením, kdy v jednu chvíli vzroste počet požadavků nebývalým způsobem oproti běžnému zatížení, na které jsou kapacity systému z důvodů efektivního využití optimalizovány. K tomuto dochází obvykle ve chvíli, kdy je systém využit pro přidělení omezeného množství lukrativních komodit. Například při on-line prodeji limitovaného počtu vstupenek na výjimečné sportovní utkání, které je dáno pouze časovým pořadím zákazníků. Rozložení takovýchto požadavků mezi více instancí v rámci systému pak může zajistit vyšší propustnost a to zejména s ohledem na ostatní uživatele systému, kteří se na přetížení vědomě nepodílí. Takové pokročilé metody jsou předmětem naší současné práce a s výhodou využívají právě metody víceúrovňových klastrů.

LITERATURA

[1] DVOŘÁK, Radek. Databáze v prostředí webu. In Sborník konference *Tvorba software 2004*. Ostrava: Tanger s.r.o., 2004, s.38–43.
[2] MOLHANEC, Martin. Kritika některých výkladů objektově orientovaného paradigmatu. In Sborník konference *Tvorba software 2004*. Ostrava: Tanger s.r.o., 2004, s.163–172.
[3] SMITH, Gordon. Oracle RAC 10g Overview. An Oracle White Paper, Oracle Corporation, November 2003.
[4] WOODACRE, M., ROBB, D., ROE, D., FEIND, K. The SGI AltixTM 3000 Global Shared-Memory Architecture. Technical whitepaper, Silicon Graphics, Inc., 2003.
[5] KŘIPAČ, Miroslav. Oracle Database & Real Application Clusters 10g on SGI Altix 350 Servers. Technická zpráva prototypového řešení, 2005.