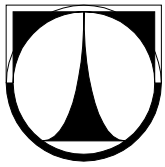
TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií



DIPLOMOVÁ PRÁCE

Víceuživatelský hlasový rozpoznávací server
s distribuovanými výpočty

Multiuser speech recognition server
with distributed computing

Liberec 2006 **Martin Pelc**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: M2612 – Elektrotechnika a informatika

Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

Víceuživatelský hlasový rozpoznávací server s distribuovanými výpočty

Multiuser speech recognition server with distributed computing

Diplomová práce

Autor: **Martin Pelc**

Vedoucí diplomové práce: Ing. Miroslav Holada, Ph.D.

Konzultant: Doc. Ing. Jan Cvejn, Ph.D.

V Liberci 18. 5. 2006

zde bude zadání diplomové práce

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/200 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě kozultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Abstrakt

Cílem diplomové práce bylo navrhnout a naprogramovat soubor aplikací představujících internetový server pro rozpoznávání izolovaných slov a krátkých frází. Požadovaný výpočetní výkon však není realizován přímo serverem, ale v síti připojenými pracovními stanicemi (distribuované výpočty). Důraz byl kladen zejména na krátkou dobu odezvy i při velkém množství současně obsluhovaných klientů. Použit byl modifikovaný rozpoznávač poskytnutý katedrou KES na TUL.

Vzniklý systém prošel testy na funkčnost rozpoznávání (výsledky poskytnuté serverem odpovídají vyřčeným slovům) a zejména zátěžovými testy (zpracování velkého množství současných požadavků). Výkonnost celého systému podle očekávání rostla lineárně s počtem připojených pracovních stanic poskytujících výpočetní výkon a průměrná doba odezvy serveru na požadavek naopak vykazovala nepřímou úměru.

Abstract

The goal of this diploma work was to design a collection of applications which would act as an intenet isolated word and short phrase recognition server. The required computing power is to be supplied by a network of workstation computers rather than the server itself (distributed computing). The whole system was designed to provide short response times even under a large number of concurrently served clients. The speech recognition engine was supplied by KES department of TUL.

The resulting system was put to a test of recognition success (results provided by the server matched the spoken words) and especially a load test (processing of a large number of concurrent recognition requests). As expected, the overall perforfamce of the server increased linearly with increasing number of connected workstations (providing computing power) and the average response time decreased accordingly.

klíčová slova: server, cluster, distribuované výpočty, distribuované rozpoznávání řeči

Obsah

Úvod

V posledním desetiletí došlo k razantnímu vývoji v oblasti počítačového rozpoznávání, zejména díky nárůstu výkonu výpočetní techniky, který umožnil v rozumném čase realizovat potřebné algoritmy na běžných počítačích. Konkrétně problematika rozpoznávání řeči, kterou se tato práce zabývá, však nenachází uplatnění pouze v aplikacích pro osobní počítače, dotýká se i menších zařízení, u kterých je žádoucí realizovat jejich ovládání pomocí přirozeného jazyka. Naneštěstí však tyto přístroje často nedisponují požadovanou výpočetní kapacitou a pamětí.

Problém hardwarového deficitu zařízení pro řešení jistých úloh (výpočtů) lze do značné míry eliminovat přenesením těchto úloh na jiné zařízení, které již potřebnými prostředky disponuje. Obecně se tento proces nazývá distribuce výpočtů, konkrétně pro hlasové rozpoznávání se vžil termín *distribuované rozpoznávání řeči* (DSR – distributed speech recognition). V drtivé většině případů se v systémech s distribuovanými výpočty používá architektura klient-server – klientská zařízení zasílají vstupní parametry úlohy serveru pomocí datového spoje, ten úlohu vyhodnotí a zašle zpět její výsledky.

Výše nastíněné řešení významným způsobem ulehčuje (potažmo zlevňuje) konstrukci klientských zařízení. Na druhou stranu však klade zvýšené nároky na hardware serveru, zejména v případě většího množství současně obsluhovaných klientů. Nejčastějším a nejjednodušším řešením pro DSR je použití jednoho či více centrálních supervýkonných serverů. Toto řešení je zpravidla finančně náročné a neflexibilní, upgrade hardwaru znamená ve většině případů výměnu celého serveru.

Tato práce se zabývá studiem možností využití sítě konvenčních počítačů (tzv. clusteru) pro potřeby DSR, k realizaci výpočetního výkonu srovnatelného se speciálními servery. Nabízí se tak možnost zlevnění a zvýšení flexibility systému.

1 Východiska a cíle práce

Práce se soustředí na distribuované rozpoznávání izolovaných slov a krátkých frází, rozpoznáváním spojité řeči se nezabývá vzhledem k řádově větší složitosti celé problematiky. Jádro celého projektu je postaveno na rozpoznávači izolovaných slov vyvíjeného na katedře KES pod pracovním označením Recon2003. Práce se též z části opírá o poznatky a závěry autorova ročníkového projektu [1], jež se zabýval problematikou tvorby, distribuce a zpracování dávkových úloh v síti počítačových stanic. S ohledem na zachování tvůrčího přístupu v celém díle, nebyl použit žádný hotový softwarový produkt na podporu clusterových výpočtů (výpočty realizované současně na více počítačích), autor vytvořil pro tento účel vlastní nástroj.

Pro vzájemnou komunikaci serveru, klientských zařízení (dále označováno jako „klient“) a pracovních stanic poskytujících výpočetní výkon (dále označováno jako „otrok“ či „slave“) byl vybrán standartní internetový protokol TCP, zejména proto, že zajišťuje exaktní pořadí doručených dat a stará se též o obnovu chyb při přenosu. Cílovou platformu tvoří operační systémy založené na jádře Microsoft Windows NT 4.0+ (MS Windows 2000/XP) s ohledem na jejich rozšířenost. Zdrojové kódy jsou psány v jazyce C++ s využitím objektů, plně anglicky okomentované.

Cílem práce bylo, aby výsledný systém vyhovoval následujícím požadavkům:

1. jedná se o soubor aplikací realizující funkci internetového rozpoznávače izolovaných slov a krátkých frází
2. rozpoznávací algoritmus vykonávají pracovní stanice, nikoli server samotný
3. server obsahuje vnitřní správu velkého množství (sto a více) připojených klientů či otroků
4. kromě samotného rozpoznávání nabízí server klientům i možnost konfigurace rozpoznávače
5. je odolný vůči výpadkům sítě, při obnovení přerušeného spojení se celý systém opět uvede v provoz bez zásahu člověka
6. jednoduše se spravuje a uvádí v činnost
7. server zahrnuje i vnitřní ochranný mechanizmus proti pádu v důsledku jeho přetížení

2 Teorie, rešerše

2.1 Distribuované výpočty

Oblast distribuovaných (přenesených) výpočtů se zabývá možnostmi využití komunikačního media k realizaci výpočtů jiným prostředkem než tím, který jejich výsledky posléze upotřebí. Pro účely této práce tím bude myšleno použití většího množství nezávislých výpočetních jednotek (zpravidla počítačů), vzájemně propojených vhodným komunikačním mediem (většinou počítačová síť), k paralelnímu provádění algoritmů náročných na výkon či spolehlivost.

V praxi se běžně vyskytují dva rozdílné přístupy k distribuovanému zpracování – lokální systémy pracující v reálném čase (tzv. clustery) a decentralizované off-line systémy (nevyžadující práci v reálném čase).

2.1.1 Clustery

Termínem cluster se zpravidla označuje skupina počítačů umístěná v jedné místnosti či objektu, propojená rychlou lokální sítí, se speciálním softwarovým vybavením pro vykonávání distribuovaných výpočtů. Jeden či více počítačů zastává řídící (organizační) roli a zbylé dodávají výpočetní výkon, pro clustery je typické nepřetržité spojení mezi počítači a kontinuální provoz. Clustery lze dále dělit podle účelu:

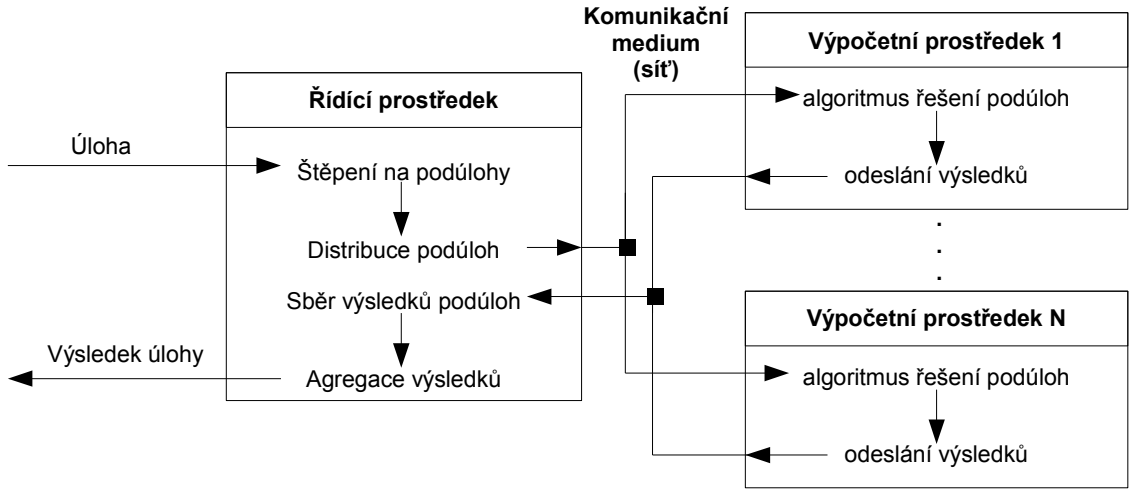
HA (high-availability) clustery – tyto slouží zejména ke zvýšení spolehlivosti systému prováděním požadovaných operací na více počítačích současně (nadbytečně) k omezení chyb vzniklých případným selháním jednotlivých počítačů. Jako příklad softwaru pro realizaci těchto clusterů lze uvést open-source projekt Linux-HA (www.linux-ha.org).

LB (load-balancing) clustery – u těchto clusterů zpravidla jeden či více řídících počítačů měří zatížení zbylých a rozhodují o rozložení vstupních požadavků tak, aby docházelo k rovnoměrnému zatěžování celého systému. Jedním z používaných softwarových řešení je projekt LVS (www.linuxvirtualserver.org).

HP (high-performance) clustery – z hlediska této práce jde o nejzajímavější způsob využití clusteru sloužící k rozkládání výpočetní náročnosti na jednotlivé počítače. Narozdíl od provádění výpočtů jedním počítačem klade distribuované

zpracování dodatečné nároky na zpracovávané úlohy – musí být rozdělitelné na nezávisle zpracovatelné podúlohy. Takové dělení však nemusí být obecně proveditelné, zejména jsou-li realizované algoritmy rekurzivní povahy (další krok algoritmu vyžaduje výsledky kroků předchozích). Metody používané pro dělení úloh jsou vždy specifické pro konkrétní účel, např. při renderování počítačových obrazů je zpravidla uplatňováno dělení scény na řádky či sektory (každý je pak zpracován jedním počítačem), vytváření videa může být rozděleno na renderování jednotlivých snímků apod. Operační systém Linux v sobě integruje podporu tohoto typu clusterů, jejich návrhem se zabývá např. projekt Beowulf (www.beowulf.org).

Princip funkce HP clusteru zjednodušeně popisuje obr. 1.



obr. 1 – schema zpracování výpočetní úlohy v HP clusteru

2.1.2 Decentralizované systémy

Tyto systémy jsou analogické clusterům, ale nejedná se o lokalizovanou a přesně vymezenou skupinu počítačů, s čímž souvisí i požadavek na maximální platformní přenositelnost použitého softwaru. Odpadá zde také nutnost nepřetržitého spojení mezi řídícím serverem a výpočetními počítači, úlohy jsou stahovány běžně pomocí internetového spojení, k jejich zpracování však nemusí dojít bezprostředně po stažení. Stejně tak i odeslání výsledků úlohy nenásleduje ihned po jejich vypočtení, celý systém nepracuje v tomto smyslu kontinuálně.

Decentralizované systémy zpravidla kombinují prvky HA a HP clusterů. Většina výpočetních stanic je anonymních, je proto nutno jednotlivé úlohy zadávat více stanicím k ověření jejich správnosti (analogické funkci HA clusteru), zároveň je zpracováváno více úloh současně (analogické funkci HP clusteru). Jednou z nejznámějších realizací decentralizovaného výpočetního systému je projekt BOINC (http://boinc.berkeley.edu).

2.2 Rozpoznávání řeči

Tato práce se zabývá rozpoznáváním izolovaných slov a krátkých fází, neřeší problematiku spojité řeči, neboť algoritmy pro její rozpoznávání vykazují rekurzivní a kontextově závislé aspekty, což významným způsobem znesnadňuje aplikaci distribuovaného zpracování (zejména dělení na nezávisle zpracovatelné podúlohy).

Počítačové rozpoznávání řeči zhruba sestává z následujících problémů:

1. **parametrizace** vstupního signálu
2. **detekce** řečové aktivity ve vstupním signálu
3. **klasifikace**

2.2.1 Parametrizace

Proces parametrizace představuje transformaci vstupní posloupnosti PCM vzorků odpovídajících nasnímanému signálu (mikrofonem) na posloupnost vektorů rozpoznávacích příznaků – uspořádaných n-tic čísel, které vhodným způsobem reprezentují obsah signálu z pohledu lidské řeči. Dále se požaduje, aby metoda parametrizace potlačovala vliv celkové síly nahrávky a stejnosměrné složky signálu na hodnoty příznakového vektoru.

Pro zachycení většiny podstatné informace obsažené v lidské promluvě postačí frekvenční rozsah do 4 kHz (přibližně rozsah telefonní linky), čemuž odpovídá vzorkovací frekcence 8 kHz při analogově-digitálním převodu. Aby nedocházelo ke zkreslení tichých nahrávek, je běžně používána 16-bitová hloubka (poskytuje dostatečný dynamický rozsah a je standardem). Rozšíření frekvenčního pásma přináší lepší reprezentaci pouze pro omezenou skupinu prvků lidské řeči (jedná se zejména o hlásky „s“ a „c“), lze jím však mírně zlepšovat úspěšnost rozpoznávače za cenu

většího objemu zpracovávaných dat (potažmo vyšší výpočetní náročnosti). Díky rostoucímu výkonu výpočetní techniky začínají systémy s větší šířkou pásma (7 kHz a více) vytlačovat systémy pracující v „telefonním“ pásmu.

V současné době nejpoužívanější metodou reprezentace řečových signálů je tzv. kepstrum. Kepstrální analýza je založena na určování krátkodobého spektra signálu a jeho transformaci sloužící k oddělení podstatné informace o hlase (odezva hlasového traktu) od nepodstatné (buzení hlasového traktu). Spektrum se měří zpravidla v úsecích o délce kolem 20 ms, přičemž se vychází z předpokladu, že řeč je složena z víceméně stacionárních úseků (povaha signálu se v nich mění jen velmi málo) odpovídajícím hláskám mluveného jazyka či přechodům mezi nimi a 20 ms je doba přibližně odpovídající nejkratší mluvené hlásce.

V současnosti jsou nejčastěji používány dvě metody výpočtu kepsta – metoda založená na výpočtu přímo ze spektra (z výsledků FFT) pomocí nelineární banky filtrů (s využitím tzv. melovské frekvence) a metoda založená na odhadu spektrální obálky signálu pomocí lineárně prediktivního modelu. Příznakový vektor získaný dříve zmíněným postupem je označován jako MFCC (Mel Frequency Cepstral Coefficients), vektor příznaků získaný pomocí lineárně prediktivního modelu pak jako LPCC (Linear Predictive Cepstral Coefficients). Algoritmy výpočtu kepstrálních koeficientů založené na LP modelování jsou obecně rychlejší za cenu snížené přesnosti, v běžné praxi se již příliš nepoužívají.

Podrobnější popis problematiky parametrizace řečových signálů lze najít ve [2] či [4].

2.2.2 Detekce řečové aktivity

Snímaný řečový signál je zpravidla souvislý, obsahuje i neřečové úseky (pomlky apod.). Úkolem detektorů (při rozpoznávání izolovaných slov) je poskytovat vybrané úseky vstupního signálu, které odpovídají jednotlivým vyřčeným slovům (popř. krátkým souvislým frázím), a odstraňovat neřečové části.

Nejjednodušší řečové detektory pracují na principu sledování energie signálu a detekci charakteristických vzestupů energie na začátku slova, respektive poklesů energie na jeho konci. Složitější detektory pak analyzují nejen úroveň signálu, ale i jeho

obsah s použitím kepstrální analýzy, kdy hledají charakteristické rysy lidské promluvy. Takové detektory jsou pak schopny odstranit i úseky vstupního signálu odpovídající různým ruchům (hlasité zvuky v pozadí), zatímco detektory zkoumající pouze energii by takové signály označily za vyřčená slova.

Více o problematice návhu řečových detektorů lze najít v [3].

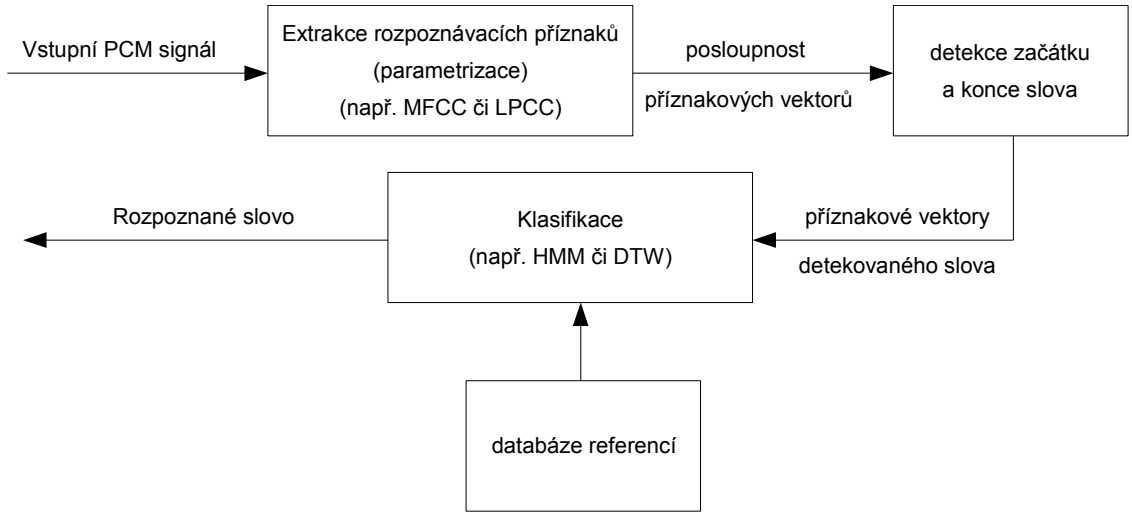
2.2.3 Klasifikace

Nejdůležitějším prvkem každého rozpoznávače je tzv. klasifikační algoritmus, který ve spolupráci s databází referencí (vzorů) tříd rozhoduje o začlenění pozorovaného neznámého objektu do jedné z těchto tříd. Při rozpoznávání izolovaných slov bývá vstupem do klasifikátoru sekvence příznakových vektorů odpovídající jednomu slovu (či frázi), klasifikátor porovná tuto sekvenci se vzory jednotlivých tříd a určí, kterému ze vzorů se vstupní posloupnost nejvíce podobá, případně zamítne zatřídění při výrazné odlišnosti od kterékoli z referencí.

V případě rozpoznávání slov bývají jednotlivými třídami právě slova, jejich soubor se zpravidla označuje jako slovník. Otázka vhodné volby referencí pro jednotlivá slova je silně závislá na zvoleném algoritmu klasifikace. Protože se jednotlivé realizace (vyřčení) téhož slova zpravidla odlišují v délce (i při vyřčení stejným jednincem), připadají v úvahu pouze algoritmy pro měření podobnosti různě dlouhých řad, uplatňují se zejména DTW (Dynamic Time Warping – dynamické borcení času) a HMM (Hidden Markov Models – skryté markovské modely). První jmenovaný je založen na principu optimálního mapování jedné řady (zpravidla referenční posloupnosti příznakových vektorů) na jinou (posloupnost příznakových vektorů pro neznámé slovo) ve smyslu minimalizace kumulované vzdálenosti (zpravidla euklidovské) mezi příznakovými vektory odpovídajících si prvků těchto řad. Skryté markovské modely využívají modelování pomocí stochastických (pravděpodobnostních) stavových automatů, každé referenci odpovídá jeden automat, míra podobnosti reference a neznámého signálu je vyjádřena pravděpodobností, s jakou příslušný automat generuje posloupnost přízankových vektorů neznámého slova. Za výsledek rozpoznávání je prohlášena ta třída, jejíž reference vykazuje nejmenší celkovou vzdálenost (u DTW) či nejvyšší pravděpodobnost (u HMM). Většina dnešních rozpoznávačů využívá právě HMM, DTW se uplatňuje již jen sporadicky.

Zbývá vyřešit otázku vytvoření referencí slov ze slovníku pro využití v klasifikačním algoritmu, přičemž se požaduje, aby byly co nejlépe reprezentovaly dané slovo v přirozeném jazyce, u systémů pro obecné použití je navíc vyžadována nezávislost na mluvčím (pohlaví, věk apod.). Zpravidla je nutno k tomuto účelu shromáždit realizace každého slova ze slovníku od co nejrůznorodější skupiny mluvčích. Pokud je však k sestavení referencí použito vždy celé slovo, dochází k technickým problémům při přidávání nových slov do slovníku (nutnost shromáždit realizace nového slova od celé skupiny mluvčích). Tento problém odstraňuje modelování vzorů slov pomocí menších stavebních kamenů přirozeného jazyka, jedná se zpravidla o tzv. fonémy (hlásky mluveného jazyka), difony a trifony (dvojice a trojice přirozených hlásek – kontextově závislé fonémy). Při tvorbě rozpoznávače stačí vytvořit databázi referencí pro tyto stavební kameny a vzor libovolného slova z nich sestavit zřetězením.

Více o problematice klasifikačních algoritmů a tvorbě referencí (proces trénování rozpoznávače) lze nalézt např. v [4]. Typický proces rozpoznávání izolovaného slova znázorňuje obr. 2.



obr. 2 – proces rozpoznávání izolovaných slov

3 Používané metody

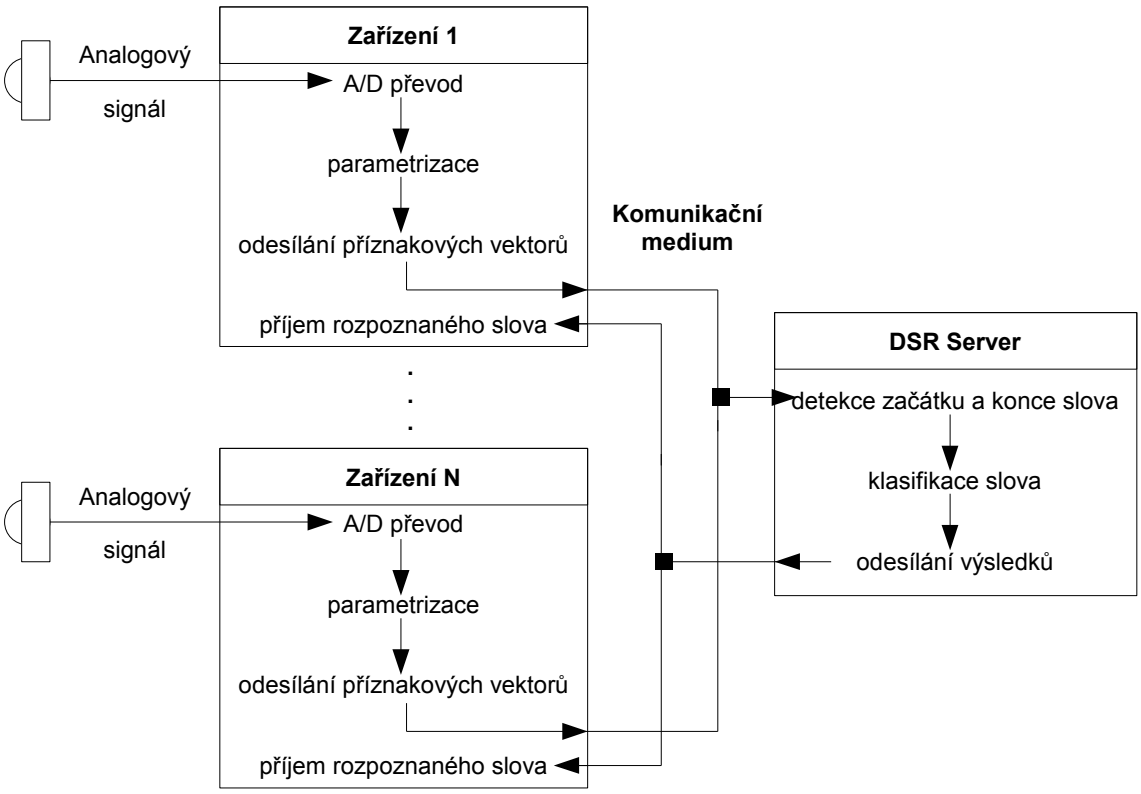
3.1 Distribuované rozpoznávání řeči

Přenesení podstatné části rozpoznávacího řetězce (obr. 2) na vzdálený počítač označujeme jako distribuované rozpoznávání řeči. Zařízení snímající lidský hlas, od kterého se očekává porozumění řeči, zpravidla provádí pouze parametrizaci (případně i jednoduchou detekci řečové aktivity), jejíž výsledky (řadu vektorů rozpoznávacích příznaků) jsou pomocí vhodného datového spoje (sítě) zasílány specializovanému serveru. Ten zrealizuje zbývající části rozpoznávacího procesu a odešle výsledky (rozpoznané slovo v textové podobě) zpět do zařízení.

Výhody tohoto přístupu spočívají zejména ve snížení hardwarových nároků na zařízení, v jednoduché modifikovatelnosti rozpoznávače (pokud se nemění algoritmus parametrizace, stačí provést update softwaru na serveru, není nutný zásah do zařízení) a v nemožnosti reverse-engineeringu samotného rozpoznávače (ten zpravidla obsahuje cenná, dlouhodobě shromažďovaná data, jež mohou být zneužita). Nevýhodou může být prodleva při síťové komunikaci či ztráta spojení, tyto problémy se však díky obecnému zvyšování kvality datových spojů vyskytují již jen omezeně.

Obr. 3 schematicky znázorňuje funkci DSR systému, v praxi bývá navíc často používána ztrátová komprese rozpoznávacích příznaků (tzv. vektorová kvantizace) k redukci datového toku komunikačním mediem, klade však dodatečné nároky na výkonnost hardwaru samotného zařízení.

V dnešní době již existuje řada serverů poskytujících služby DSR, jeden z nich provozuje i TUL pod názvem DUNDIS (http://dundis.kes.tul.cz). Komerční nasazení DSR se předpokládá v mobílních sítích 3. generace (3G), konkrétní řešení již nabízí např. firma IBM (www.haifa.ibm.com/projects/multimedia/dsr/index.html). O standardizaci komunikace mezi klientským zařízením a serverem v DSR usiluje skupina STQ-Aurora v rámci organizace ETSI (http://portal.etsi.org/stq).



obr. 3 – příklad uspořádání a funkce DSR systému

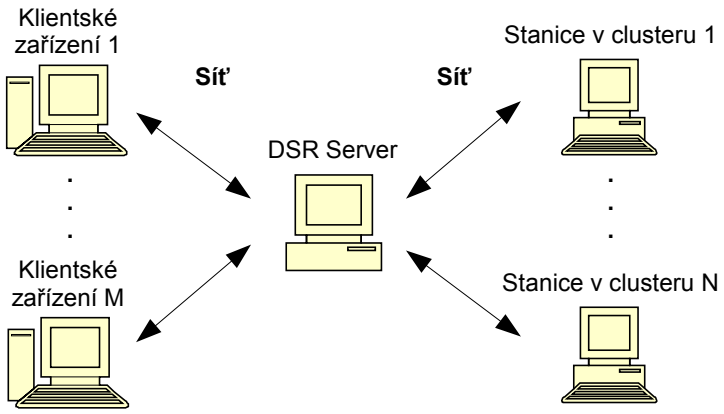
3.2. DSR a clusterové výpočty

Autorovi není známo, že by se již nějaký projekt zabýval otázkou aplikace clusterů při realizaci DSR systémů. V oblasti rozpoznávání řeči jsou clustery často používány pro trénování a testování různých verzí rozpoznávačů, kdy je zapotřebí rychle zpracovávat obrovské databáze řečových záznamů, samotný provoz rozpoznávače se ale zatím na clusterech neprovádí.

4 Vlastní řešení

Základní ideu uspořádání DSR systému s clusterovými výpočty vystihuje obr. 4. Před započetím vlastní realizace bylo nutno vyřešit následující **základní problémy:**

1. návrhnout vhodné postupy a mechanizmy pro efektivní realizaci clusterových výpočtů a tyto naprogramovat
2. vymyslet vhodný mechanizmus dělení úlohy rozpoznávání na izolované (elementární) podúlohy zpracovatelné v clusteru
3. úpravit výchozí rozpoznávač – zejména separovat jeho funkční bloky tak, aby mohly být provozovány nezávisle, na různých počítačích



obr. 4 – ideové uspořádání navrhovaného DSR systému

Řešení výše zmíněných problémů pak tvoří základní kameny pro finální syntézu zamýšleného DSR systému.

Po dohodě s vedoucím projektu byly stanoveny následující konvence. Cílovou platformu tvoří operační systémy založené na jádře Microsoft Windows NT 4.0 a novější (Windows 2000/XP). Komunikace v síti je postavena na protokolu TCP, funkce realizovány pomocí MS Windows Sockets 2.0 (API). Vývoj softwaru probíhal v prostředí MS Visual C++ 6.0 (narozdíl od novějších prostředí nabízí bohatou podporu API) s využitím objektového programování. Při vývoji aplikací nebyl použit kód třetích osob, pouze prostředky API a katedrou poskytnutý rozpoznávač.

4.1 Clusterové výpočty

Úloha realizace clusterových výpočtů byla formulována následovně – k dispozici je „n + 1“ počítačů propojených v síti (obecně se mohou lišit hardwarem), dále je dáno „m“ úloh (obecně mohou být různě výpočetně náročné) a algoritmus jejich řešení. Jak nejrychleji vyřešit tyto úlohy s použitím daných počítačů?

4.1.1 Teoretický návrh

Navržený postup je následující – právě jeden z počítačů zastupuje řídící roli (bude označován jako *master – pán*; přikazuje ostatním, co mají dělat), zbylé počítače (označovány jako *slave – otrok*; vykonávají všechnu namáhavou práci) se k němu připojují. Master zabezpečuje přebírání úloh od zadavatele, jejich distribuci slave počítačům, sběr výsledků a jejich předávání zadavateli. Připojené slave počítače čekají na zadání úlohy masterem, realizují algoritmus řešení a odesílají zpět výsledky. Vnitřní organizace „práce“ je postavena na systému front – fronta (nezpracovaných) úloh, fronta volných (nepracujících) slave počítačů a fronta zaměstnaných slave počítačů.

Životní cyklus slave počítače z pohledu mastera:

1. Inicializace – po připojení k masterovi je otroku zaslána série úvodních úloh sloužících ke vzájemné synchronizaci (je-li tato potřeba); do doby, než otrok zpracuje inicializační úlohy, je zařazen do fronty zaměstnaných otroků
2. Otrok nepracuje – je zařazen na konec fronty nepracujících otroků a čeká, až „přijde na řadu“ a bude mu zadána nová úloha
3. Otrok přišel na řadu – otrok se posunul na první místo ve frontě nepracujících otroků; pokud není fronta úloh prázdná, pokračuje otrok krokem č. 4; pokud nemá master úlohu ke zpracování, čeká otrok na její příchod
4. Otrok pracuje – otrok přišel na řadu, master vyzvedl čekající úlohu z fronty a přidělil ji otrokovi, ten je zařazen do fronty pracujících
5. Otrok dokončil práci – byl zrealizován výpočetní algoritmus a otrok odeslal zpět výsledky, je odstraněn z fronty pracujících a pokračuje dále v činnosti krokem 2, výsledky úlohy master předá spolu s jejím zadáním zadavateli

6. Dojde-li v průběhu cyklu k přerušení spojení a otrok je ve frontě nepracujících, dojde k jeho odstranění z této fronty. Pokud pracoval v době odpojení na úloze (jiné než inicializační), je tato zařazena na první místo ve frontě úloh (bude při první příležitosti opět zpracována) a otrok je odstraněn z fronty pracujících.

Všechny nově příchozí úlohy jsou řazeny na konec fronty úloh a čekají na zpracování. Krok 6 životního cylku otroka též zajišťuje, že případné výpadky slave počítačů z clusteru výrazným způsobem neovlivní zpracování úloh.

4.1.2 Implementace

Master (určen k běhu na řídícím počítači) a slave (určen k běhu na otrockých počítačích) jsou realizovány jako autonomní objekty jazyka C++ (třídy *CRFMaster* a *CRFSlave*, soubory *rf2.h* a *rf2.cpp*), využívají asynchronní komunikace po síti za pomoci vlastního vlákna. Třídy jsou napsány obecně, mohou sloužit pro realizaci libovolných clusterových výpočtů. Algoritmus zpracování úloh a předávání jejich výsledků jsou řešeny jako volání virtuálních callback metod těchto objektů, ty je možno libovolně definovat odvozením potomka třídy a jejich převálcováním (override).

V objektu master jsou pro připojené otroky a přijaté úlohy vytvářeny příslušné kontejnerové objekty (*CRFMConnection* a *CRFMBatch*, soubory *rf2.h* a *rf2.cpp*). Fronty jsou realizovány jako dvojcestné spojové seznamy těchto kontejnerových objektů, v principu tedy není délka front omezena.

Oproti teoretickému návrhu vyvstaly při realizaci ještě následující problémy:

1. Při přerušení TCP spojení není vždy na toto upozorněna aplikace – tento jev nastává při hardwarových poruchách, zejména při fyzickém přerušení síťové cesty či neočekávaném vypnutí vzdáleného systému (odpojení od elektřiny)
2. Při nedostatečné výpočetní kapacitě připojených otroků může převýšit frekvence nově zadávaných úloh rychlost jejich řešení – úlohy se hromadí ve frontě mastera, což může vést k vyčerpání volné paměti počítače a pádu operačního systému
3. Může dojít k přerušení činnosti otroka v průběhu zpracování dávky

První problém byl vyřešen zavedením tzv. *heartbeat* mechanizmu. Master zasílá nepracujícím otrokům v pravidelných intervalech (5s) speciální povel (heartbeat), kterým dává otrokům najevo, že spojení nebylo přerušeno. Sám master zjistí přerušení spojení tím, že selže příslušný API příkaz při odeslání heartbeat povelu. Pokud otrocký objekt neobdrží delší dobu tento povel (15s) zavolá příslušnou callback metodu, kterou dá aplikaci najevo ztrátu spojení.

Problém číslo dvě není implicitně objektem master řešen, lze však kdykoli zjistit celkový počet připojených otroků a celkový počet úloh, které byly zadány, ale nebyly ještě vyřešeny. Zadavatel úloh tedy může diagnostikovat míru přetížení systému a podle potřeby pozastavit vkládání nových úloh do clusteru.

Posledně zmíněný problém byl vyřešen možností zadání tzv. *time-out*u pro každou úlohu. V případě uplynutí této lhůty je otrok považován za „mrtvého“ a je na něj hleděno, jako by došlo k přerušení spojení (aplikuje se bod 6 životního cylku). Aby se zamezilo „zničení“ clusteru v případě zadání úlohy s velmi krátkým time-outem (žádný z otroků by nedokázal úlohu vyřešit v požadovaném čase a daná úloha by způsobovala odpojení všech otroků), byl stanoven toleranční práh pro každou úlohu. Po třech neúspěšných pokusech o vyřešení úlohy je úloha zamítnuta (zadavateli je vráceno její zadání bez řešení).

4.2 Dělení úlohy rozpoznávání

Clusterového zpracování předpokládá řešení izolovaných (ucelených) úloh. Při pohledu na rozpoznávací řetězec (obr. 2) je klíčovým prvkem detektor slov, neboť všechny datové toky před ním mají proudovou (kontinuální) povahu a nelze je tudíž uměle dělit na nezávislé bloky (samotný detektor je kontextově závislý) bez narušení celého procesu rozpoznávání. Naopak data vystupující z detektoru již mají povahu blokovou, jedná se o izolované řady vektorů rozpoznávacích příznaků odpovídajících jednotlivým slovům či krátkým frázím, víceméně nezávislé na ostatních detekovaných slovech. Této vlastnosti (převodu proudu dat na kontextově nezávislé úseky) bylo s výhodou využito pro vytváření úloh a jejich clusterové zpracování.

Je-li úloha rozpoznání jednoho slova přidělena vždy jednomu počítači z clusteru, nedochází sice k urychlení rozpoznávání jednotlivých slov, jedná se však o mocný

nástroj pro současné zpracování velkého množství paralelních proudů (současně prováděných rozpoznávání). Snaha o urychlení úlohy rozpoznání jednoho slova je též neopodstatněná vzhledem k faktu, že tuto úlohu vyřeší běžný počítač (otrok z clusteru) řádově v desetinách sekundy (v závislosti na velikosti slovníku a algoritmu klasifikace). Za úlohu, kterou je nutno rozdělit na podúlohy ve smyslu clusterového zpracování, můžeme tudíž považovat rozpoznávání většího množství paralelních proudů (nikoli signálu z jediného zdroje).

Aby byla úloha ucelená a nezávisle zpracovatelná, je nutno k příznakovým datům detekovaného slova přidat i kompletní konfiguraci rozpoznávače.

4.3 Úprava rozpoznávače

4.3.1 Výchozí stav rozpoznávače

Rozpoznávač použitý v projektu byl vyvinut na katedře KES (při TUL) Prof. Janem Nouzou, nese pracovní označení Recon2003. Jedná se o kompaktní rozpoznávač používající signál ze zvukové karty počítače. Klíčové parametry systému:

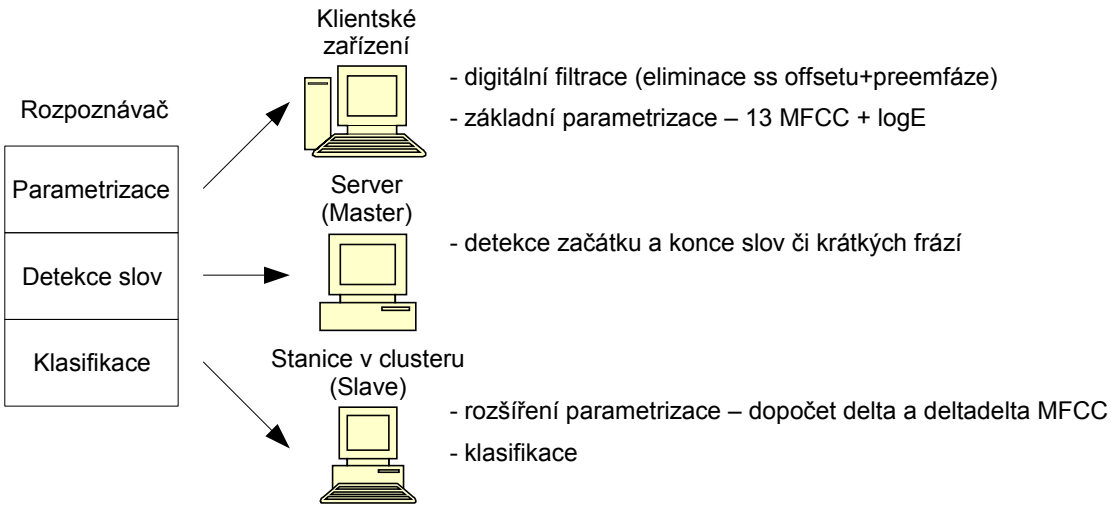
- formát vstupních dat – 8 kHz monofonní, 16-bitový PCM signál
- rámce pro parametrizaci o délce 25 ms, vzájemný posun rámců 10 ms
- parametrizace MFCC – 13 kepstrálních koeficientů + odpovídající delta a deltadelta příznaky (aproximace první a druhé derivace těchto koeficientů) + logaritmus energie rámce (používá se pro detekci začátku a konce slova)
- detektor založený na sledování vývoje energie signálu, adaptující se na úroveň šumu (ruchu) v neřečových částech
- klasifikace založena na multimixturových markovských modelech, reference slov ze slovníku jsou sestavovány řetězením modelů jednotlivých fonémů slova
- slovníky a fonémové modely jsou uloženy ve zvláštních souborech a jsou načítány při inicializaci rozpoznávače

4.3.2 Cílový stav rozpoznávače

Cílem práce je především přenesení maxima výpočetní náročnosti rozpoznávání do clusteru, čemuž odpovídá i navržené rozložení jednotlivých částí rozpoznávače

(obr. 5). K redukci datových toků sítí byl výpočet delta a deltadelta MFCC odložen až před samotnou klasifikaci (výpočet provádí počítače v clusteru).

Jednotlivé funkční celky rozpoznávače byly separovány a přeprogramovány do objektových tříd – záznam zvuku a parametrizace (*CSoundRecord* v souboru *recon2003mp.h* a *.cpp*), detektor slov (*CSpeechDetector* v *recon2003mp.h* a *.cpp*), rozpoznávač (*CSRRecognizer* v *srfiles.h* a *.cpp*) a pomocné soubory rozpoznávače (*CSRDictionaryFile* – soubor slovníku, *CSRPhonemFile* – soubor fonetických modelů a *CSRAplhaFile* – soubor pro kódování abecedy, vše definováno v *srfiles.h* a *.cpp*).



obr. 5 – rozložení funkčních bloků rozpoznávače v navrhovaném DSR systému

Zdrojové soubory obsahující kód poskytnutý katedrou ani datové soubory obsahující fonetické modely nebudou umístěny na přiloženém CD. Jsou výsledkem dlouholetého vývoje a není žádoucí jejich zveřejnění.

4.4 Syntéza konečného řešení

Po vyřešení většiny základních problémů zbývá přistoupit k realizaci (naprogramování) konečných aplikací. Schema jejich společné činnosti je v příloze A.

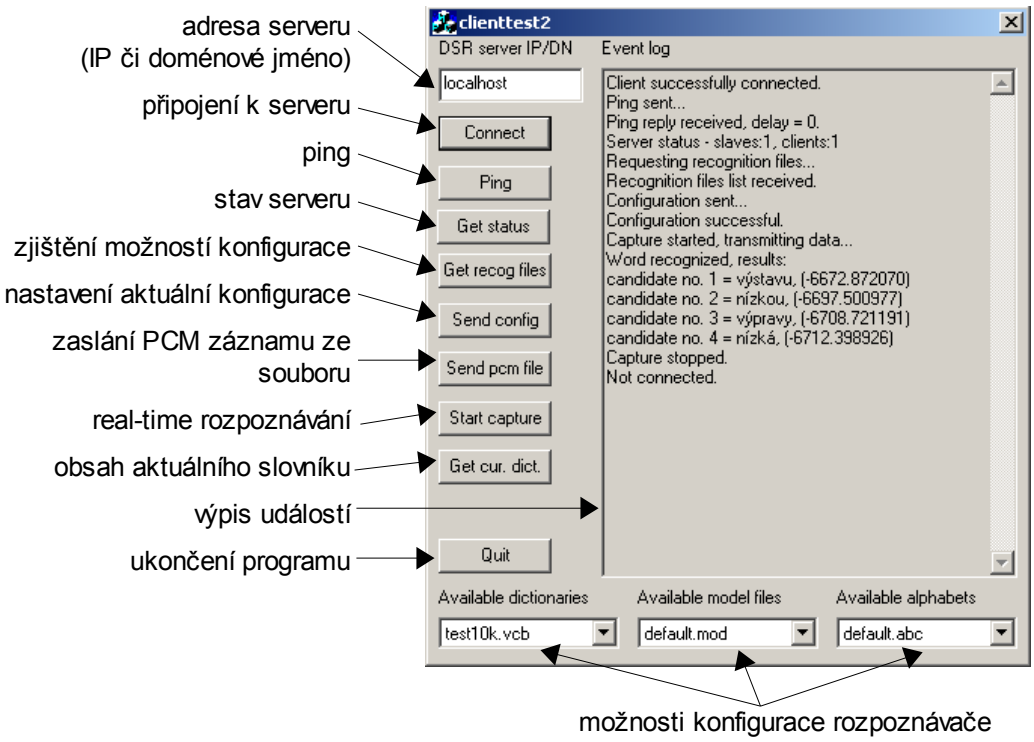
4.4.1 Klient

Základní úloha klienta v celém řetězci spočívá v získávání digitálních zvukových dat, jejich parametrizaci a odesílání na server. Mimo tuto nezbytně nutnou činnost má

také možnost konfigurovat rozpoznávač, resetovat stav detekoru slov a zjišťovat další informace od serveru (počet slov doposud detekovaných v proudu rozpoznávacích příznaků, možnosti konfigurace, výpis slov z aktuálního slovníku, počet připojených klientů či počítačů v clusteru a měření síťové prodlevy – ping).

Schema síťové komunikace, serverem nabízené funkce a formát datových toků mezi klientem a serverem jsou popsány ve 4.2.2.

Pro demonstrační účely a k testování serveru vznikl softwarový klient ve formě objektové třídy jazyka C++ (*CDSRClient* v souboru *dsr2.h* a *.cpp*). Tento byl poté integrován do ukázkové aplikace (*clienttest2* – obr. 6), na které byly testovány jednotlivé funkce nabízené serverem, zejména správná funkce rozpoznávače. K serveru se připojuje na portu 2030, který byl vybrán jako implicitní pro připojování klientů.



obr. 6 – testovací klientská aplikace

Po připojení klienta k serveru je možno získat seznamy souborů rozpoznávače (slovníky, hláskové modely a abecedy). Testovací klient je zobrazuje v dolní části uživatelského rozhraní, umožňuje v nich též vybírat. Vybraná konfigurace se odesílá na server volbou *Send config*, přičemž jsou k vybranému slovníku přidány tři

uživatelské fráze („nové slovo“, „další slovo“ a „přidané slovo“ - slouží k otestování funkce přidávání uživatelských slov do předdefinovaných slovníků) a počet zpět zasílaných kandidátů (výsledků rozpoznávání) je nastaven na čtyři.

Klient umožňuje zachytávání zvuku a rozpoznávání v reálném čase (tlačítko *Start capture*), popř. lze zasílat nahrávky ze souboru (tlačítko *Send pcm file*). Výsledky rozpoznávání slova obdržené ze serveru, stejně jako všechny události, jsou vypisovány do vyhrazené části grafického rozhraní.

4.4.2 Server

V navrhovaném systému plní server trojí roli – zabezpečuje komunikaci s klienty (vyřizuje jejich požadavky), realizuje funkci detektoru slov a zároveň slouží jako řídící počítač pro výpočetní cluster. Použitím modifikovaného autonomního systému navrženého v bodě 4.1 se úloha serveru vzhledem ke clusteru redukuje na roli zadavatele výpočetních úloh. Dále zbývá vyřešit problematiku správy a obsluhy připojených klientů.

4.4.2.1 Spouštění serveru

Server nabízí jisté možnosti konfigurace při jeho spouštění, jedná se zejména o nastavení:

- IP adresa a port pro připojení klientů – pokud není adresa explicitně zadána, server při startu zjistí seznam adres dostupných pro počítač a použije první ze seznamu; výchozí hodnota portu pro klientská připojení byla autorem stanovena na 2030
- IP adresa a port pro připojení otrockých počítačů (clusteru) – pokud není specifikována, použije se první zjištěná lokální IP adresa; výchozí hodnota portu pro připojení ke clusteru je 2020
- Práh přetížení clusteru – pokud přesáhne počet čekajících úloh v clusteru na jednoho otroka (podíl počtu čekajících úloh a počtu připojených otroků) tuto hodnotu, přepne se server do módu, kdy nezařazuje nově detekovaná slova k rozpoznávání v clusteru (zahazuje je, klienty na tuto skutečnost upozorní); tento mód trvá až do vyprázdnění clusteru (počet čekajících úloh na jednoho

otroka je menší než jedna), hystereze je zavedena z důvodu snížení frekvence přepínání do tohoto módu při přetěžování serveru

- Adresář se soubory rozpoznávače – při spuštění server načte všechny soubory rozpoznávače z tohoto adresáře do paměti, ty pak používá pro synchronizaci otroků (zajití jim tak stejnou schopnost rozpoznávání) a jejich seznam tvoří možnosti pro konfiguraci rozpoznávače (viz 4.4.1, konfigurace rozpoznávače v klientovi); soubory jsou identifikovány podle charakteristických přípon (*vcb* – soubory slovníků, *mod* – soubory hláskových modelů a *abc* – soubory abecedy), server vyžaduje v adresáři přítomnost výchozích souborů každého typu (*default.vcb*, *default.mod* a *default.abc*)

Zavedení podmínky existence výchozích souborů rozpoznávače vyřešilo zejména následující problémy – zajištění funkčnosti rozpoznávače (ten vyžaduje pro svou činnost alespoň jeden soubor od každého typu) a výchozí konfigurace rozpoznávače (pokud klient začne rozpoznávat, aniž by zaslal vlastní konfiguraci). V případě neexistence některého z výchozích souborů selže pokus o spuštění serveru.

Pokud dojde ke změně souborů ve specifikovaném adresáři poté, co byl již server spuštěn, tato modifikace se serveru nijak nedotkne (soubory již byly načteny do paměti). K uplatnění změn je třeba server restartovat.

Po úspěšném spuštění server naslouchá příchozím požadavkům na spojení. Během běžné činnosti serveru neexistuje žádná akce, při které by server svévolně ukončil svoji činnost (ukončí ho zpravidla operační systém či správce).

4.4.2.2 Připojení a odpojení otroka

Jak již bylo zmíněno výše, server používá pro realizaci clusteru modifikovanou verzi clusterového softwaru z 4.1. Přidána byla zejména inicializace otroka ve smyslu zaslání všech souborů rozpoznávače, čímž je zajištěn soulad v možnostech konfigurace rozpoznávače. Další činnost otroka poté odpovídá životnímu cyklu otroka podle 4.1.1. Úlohy, které otrok řeší jsou izolovanými a celistvými úlohami rozpoznání slova či krátké fráze (zadání úlohy obsahuje sekvenci rozpoznávacích příznaků jednoho slova a kompletní konfiguraci rozpoznávače včetně případné uživatelské části slovníku).

Postup při přerušení spojení s otrokem se plně shoduje s bodem 6 jeho životního cyklu, jak byl nastíněn v 4.1.1.

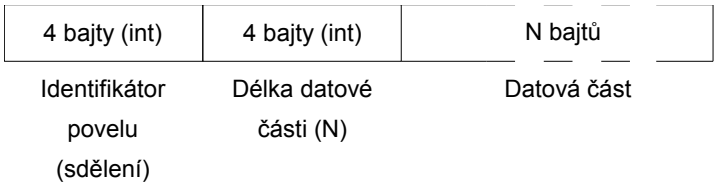
4.4.2.3 Připojení a odpojení klienta

Po připojení klienta je na serveru vytvořena příslušná datová struktura pro práci s ním (objekt umožňující přijímání a zasílání zpráv přes síť, obsahující konfiguraci rozpoznávače specifickou pro daného klienta) a je mu vytvořen a přiřazen jeho vlastní detektor slov. Server poté s klientem komunikuje dle bodu 4.4.2.4.

V případě odpojení klienta jsou na serveru uvolněny příslušné datové struktury spolu s detektorem. Slova, která byla detekována v proudu řečových příznaků od klienta a která čekají v clusteru na rozpoznání, nejsou z clusteru odstraňována (došlo by k narušení průhledného principu jeho fungování), pouze po obdržení výsledků rozpoznávání z clusteru tyto server zahazuje.

4.4.2.4 Funkce poskytované serverem

V komunikaci mezi serverem a klientem přes TCP spojení je používáno datové schema z obr. 7. Protože je server a komunikace s ním nejdůležitější částí práce, bude do detailu rozebrána. Její popis by měl umožnit (nebo alespoň přiblížit) čtenáři tvorbu vlastních klientů (i na jiných platformách než PC), současně též poskytuje hlubší náhled na funkce serveru.



obr. 7 – formát zpráv při klient-server komunikaci

Povel: PING
Identifikátor: 0x00000002
Směr: klient→server
Význam: měření celkové prodlevy komunikace („tam a zpět“)
Datová část: libovolná, klient do ní zpravidla uloží informaci o čase vyslání zprávy
Funkce serveru: server zašle přijatou zprávu zpět v nezměněné podobě (stejný identifikátor i datová část)

Směr: server→klient

Význam: odpověď serveru na zprávu PING obdrženou od klienta

Datová část: shodná s datovou částí PING zprávy klienta

Užitek pro klienta: klient zpravidla vypočte rozdíl mezi aktuálním časem a časem zakódovaným ve zprávě, ten je hodnotou síťové prodlevy

Povel: GET STATUS

Identifikátor: 0x00000008

Směr: klient→server

Význam: žádost o zaslání aktuálního stavu serveru

Datová část: prázdná, nulové délky

Funkce serveru: server zašle zpět informace o počtu k serveru připojených klientů, otroků a indikátor přetížení serveru (ano/ne)

Směr: server→klient

Význam: odpověď serveru na zprávu GET STATUS obdrženou od klienta

Datová část: 4B (int) počet klientů, 4B (int) počet otroků, 4B (bool) přetížení

Užitek pro klienta: klient může poskytnutou informaci využít k diagnostickým účelům, vizualizaci apod.

Povel: GET FILE LIST

Identifikátor: 0x00000004

Směr: klient→server

Význam: žádost o zaslání seznamu souborů rozpoznávače (představující možnosti konfigurace rozpoznávače)

Datová část: prázdná, nulové délky

Funkce serveru: server zašle zpět seznam souborů pro rozpoznávání, které při svém startu načetl z příslušného adresáře

Směr: server→klient

Význam: odpověď serveru na zprávu GET FILE LIST obdrženou od klienta

Datová část: pole následujících struktur
64B (char[64]) název souboru – nulou ukončený (ANSI) řetězec
4B – bez významu
4B (int) typ souboru – 0=abeceda, 1=phonet. modely, 2=slovník

Délka dat: násobek velikosti struktury (72B)

Funkce klienta: vydělením délky datové části velikostí struktury získá klient počet zaslaných struktur, názvy souborů poté klient používá ke konfiguraci rozpoznávače

Povel: CONFIG

Identifikátor: 0x00000003

Směr: klient→server

Význam: nastavení parametrů rozpoznávání

Datová část: 64B (char[64]) název souboru abecedy
64B (char[64]) název souboru s fonetickými modely
64B (char[64]) název souboru slovníku
4B (int) počet serverem vracených kandidátů při rozpoznání
4B (int) velikost uživatelského slovníku v B (včetně oddělovačů)
4B – ignorováno

<nepovinné> uživatelský slovník – nulou oddělená slova uživatelského slovníku, ukončeno dvěma nulami

Délka dat: 204B + velikost uživatelského slovníku (0 když není zadán) max. velikost uživ. slovníku je 20KB

Funkce serveru: server zkontroluje, zda jsou požadované soubory k dispozici a nastaví konfiguraci rozpoznávače (pro úlohy zasílané do clusteru) v případě jejich existence, pošle zpět zprávu o výsledku operace

Směr: server→klient

Význam: indikace úspěchu či chyby při pokusu klienta o konfiguraci

Datová část: 4B (int) návratová hodnota – 0=úspěch, -86=špatná struktura konfigurační zprávy (neodpovídají velikosti), -88=příliš velký uživatelský slovník (20KB max.), -85=požadovaný soubor rozpoznávače není přítomen na serveru

Povel: RECOG DATA

Identifikátor: 0x00000005

Směr: klient→server

Význam: klient zasílá rozpoznávací příznaky na server

Datová část: 4B (int) počet zasílaných framů (vektorů) - N
(4*N)B (float[N]) pole logaritmů energií zasílaných framů
(4*13*N)B (float[N][13]) pole N vektorů kepstrálních příznaků, uspořádání c1, c2, … c12, c0

Délka dat: (14xN + 4)B

Funkce serveru: server přidá přijatá data do detektoru klienta; v případě, že po přidání těchto dat detektor našel v doposud přijatých datech celé slovo, vyzvedne se sekvence příznaků tohoto slova z detektoru a vygeneruje se úloha rozpoznání pro cluster, inkrementuje se počítadlo detekovaných slov

Směr: server→klient

Význam: cluster dodal výsledky rozpoznávání slova detekovaného v proudu příznakových dat od klienta, server zasílá tyto výsledky klientovi

Datová část: pole následujících struktur
40B (char[40]) kandidát na výsledek rozpoznávání (slovo ze slovníku) – nulou ukončený ANSI řetězec
4B (float) rozpoznávací skóre kandidáta

Délka dat: násobek délky struktury (44B)

Funkce klienta: počet vrácených kandidátů lze zjistit podělením délky datové části velikostí struktury; nejlepší kandidát (rozpoznané slovo) je v poli první, další jsou sestupně uspořádáni podle skóre

Povel: RESET DETECTOR

Identifikátor: 0x00000006

Směr: klient→server

Význam: požadavek na resetování detektoru a počítadla slov

Datová část: žádná, nulová délka

Funkce serveru: klientův detektor je na serveru vynulován, jsou zahozena všechna řečová data zbývající v detektoru, je vynulováno

počítadlo detekovaných slov

Směr: server→klient

Význam: žádný, server neoznamuje úspěšný reset detektoru klientům

Povel: GET WORD COUNT

Identifikátor: 0x00000007

Směr: klient→server

Význam: žádost, aby server zaslal počet doposud detekovaných slov

Datová část: žádná, nulová délka

Funkce serveru: server zašle zpět počet slov, která byla doposud detekována v proudu řečových dat od klienta a zresetuje počítadlo

Směr: server→klient

Význam: server zaslal počet doposud detekovaných slov a zresetoval čítač

Datová část: 4B (int) počet detekovaných slov

Užitek pro klienta: zejména při rozpoznávání ze souborů s nahrávkou, ke zjištění počtu slov obsažených v nahrávce; při rozpoznávání v reálném čase postrádá smysl

Povel: GET DICTIONARY

Identifikátor: 0x00000009

Směr: klient→server

Význam: požadavek o zaslání výpisu slov z aktuálně vybraného slovníku

Datová část: žádná, nulová délka

Funkce serveru: server extrahuje slova ze slovníku, který je pro daného klienta aktuálně nastaven v konfiguraci rozpoznávače (včetně případných uživatelských slov), a pošle je klientovi

Směr: server→klient

Význam: server zasílá požadovaný výpis slovníku

Datová část: nulami oddělená slova ze slovníku, data ukončena dvěma nulami

Délka dat: celková délka dat včetně oddělovacích a ukončovacích nul

Užitek pro klienta: pro diagnostické účely, k ověření výskytu slova ve slovníku apod.

Povel: HEARTBEAT

Identifikátor: 0x0000000A

Směr: klient→server

Význam: žádný, server tuto zprávu nepřijímá (nezná)

Směr: server→klient

Význam: server testuje každých 5s spojení s klientem

Datová část: žádná, nulová délka

Užitek pro klienta: zanedbatelný, může sloužit k indikaci připojení k serveru

Povel: OVERLOADED

Identifikátor: 0x0000000B

Směr: klient→server

Význam: žádný, server tuto zprávu nepřijímá (nezná)

Směr: server→klient

Význam: server ohlašuje zamítnutí rozpoznání detekovaného slova

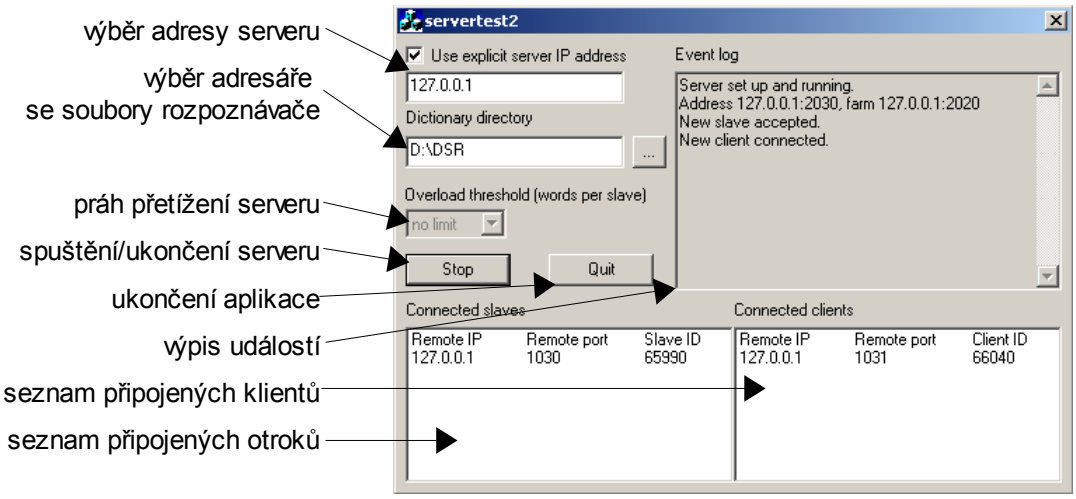
z důvodu přetížení clusteru (slovo je zahozeno)
Datová část: žádná, nulová délka
Užitek pro klienta: klient může indikovat uživatelovi dočasnou nefunkčnost rozpoznávání z důvodů přetížení systému

Povel: UNKNOWN
Identifikátor: 0xFFFFFFFF
Směr: klient→server
Význam: žádný, server tuto zprávu nepřijímá (nezná)
Směr: server→klient
Význam: server oznamuje přijetí klientské zprávy s neznámým identifikátorem
Datová část: žádná, nulová délka
Užitek pro klienta: pro testovací účely; klient zjistí, že server odeslané zprávě neporozuměl

4.4.2.5 Výsledná aplikace

Aplikace plnící funkci serveru byla vyhotovena ve dvou variantách – testovací varianta s grafickým uživatelským rozhraním (*servertest2.exe*) a verze pro běžný provoz ve formě služby systému Windows bez uživatelského rozhraní (*DSRServer.exe*). Server ve formě služby je konfigurován a zaváděn do operačního systému pomocí zvláštní aplikace (*DSRServerSetup.exe*).

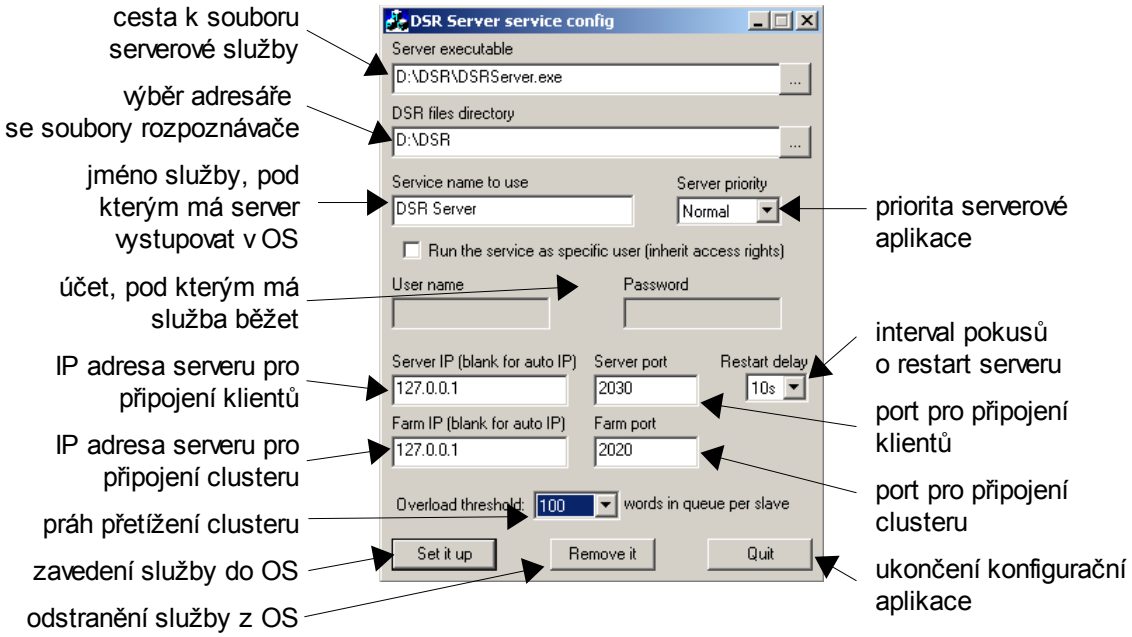
Rozhraní testovací aplikace ukazuje obr. 8, to umožňuje veškerá nastavení s výjimkou portů, kterým jsou přiřazeny výchozí hodnoty (2030, resp. 2020). Zobrazují se na něm výpisy důležitých událostí a seznamy připojených klientů a počítačů z clusteru. Detailní záznam všech akcí a událostí (i chyb) je zapisován do souboru *serverlog.txt*, který je vytvářen při spuštění serveru v pracovním adresáři aplikace. Účelem této aplikace bylo primárně ladění funkčnosti serveru, lze ji však s výhodou použít pro diagnostiku chyb při neúspěšném zavádění verze serveru bez uživatelského rozhraní či při problémech s jeho provozem.



obr. 8 – testovací serverová aplikace

Na obr. 9 je grafické rozhraní konfigurační aplikace sloužící k zavádění serveru jako služby operačního systému. Oproti testovací aplikaci obsahuje verze pro běžný provoz též detekci změn v seznamu lokálních IP adres (signalizuje odpojení od sítě či neplatnost použité IP adresy). Když změna nastane, je server automaticky restartován a pokud se stala předtím používaná IP adresa neplatnou (a tudíž nepoužitelnou), restart serveru neuspěje a aplikace ho opakuje v pravidelných intervalech (nastavitelných položkou *Restart delay*). Pokusy o restart serveru se dějí i v případě, že se nepovede start z jiné příčiny, např. chybí-li v adresáři se soubory rozpoznávače výchozí (default) soubory. Pro služby systému Windows bohužel neexistuje žádný způsob výstrahy uživatele (zejména z důvodu běhu pod jiným uživatelským účtem, zpravidla *SYSTEM*), doporučuje se tedy před zavedením služby odzkoušet start serveru testovací aplikací.

Pokud není při zavádění serverové služby zadán uživatelský účet, služba je spouštěna pod standardním systémovým účtem (*SYSTEM*). Pod zadaným jménem služby lze posléze server najít a zastavit či restartovat ve standardních prostředcích pro správu OS – skupina *Služby*. Zavedení serveru se provádí stiskem tlačítka „*Set it up*“, přičemž dojde k registraci do služeb systému a k vytvoření souboru s konfigurací v adresáři s *exe* souborem serveru, pod stejným jménem ale s příponou *ini* – ten je serverem automaticky načítán při startu služby. Důležité události při běhu (např. chyby) jsou zaznamenávány do tzv. logovacího souboru, ten server vytváří v adresáři s *exe* souborem, pod stejným názvem ale s příponou *log*.



obr. 9 – konfigurační aplikace serveru jako služby OS

V případě potřeby odstranění serveru ze systému stačí v konfigurační aplikaci zadat jméno služby, pod kterým byl zaveden, a zvolit „*Remove it*“. Dojde tak k zastavení jeno činnosti a odebrání ze systému (nejsou však smazány žádné soubory).

4.4.3 Slave

4.4.3.1 Princip činnosti

Software pro počítače v clusteru vychází z návrhu v 4.1. Zbývalo jen přidat samotný algoritmus řešení úlohy klasifikace zadávané serverem a dořešit synchronizaci.

Jak již bylo zmíněno v bodu 4.4.2.2, zasílá vždy server otroku kompletní sadu souborů rozpoznávače (názvy + jejich obsah), které jsou následně používány při rozpoznávání. Otrok je po jejich přijetí uchovává pouze v operační paměti, navíc v reorganizované podobě, je tak značně redukována možnost jejich získání z počítače clusteru (lze ho tedy provozovat bez větších nároků na zabezpečení, např. v počítačové učebně).

Po provedení synchronizace je již otrok využíván clusterem stadnardním způsobem. Přijímané úlohy jsou vždy ve tvaru – vektorové příznaky slova + názvy souborů, jež mají být použité při rozpoznávání + (nepovinně) seznam uživatelských

slov (uživatelské rozšíření vybraného fixního slovníku). Tuto úlohu pak lze považovat za nezávisle řešitelnou.

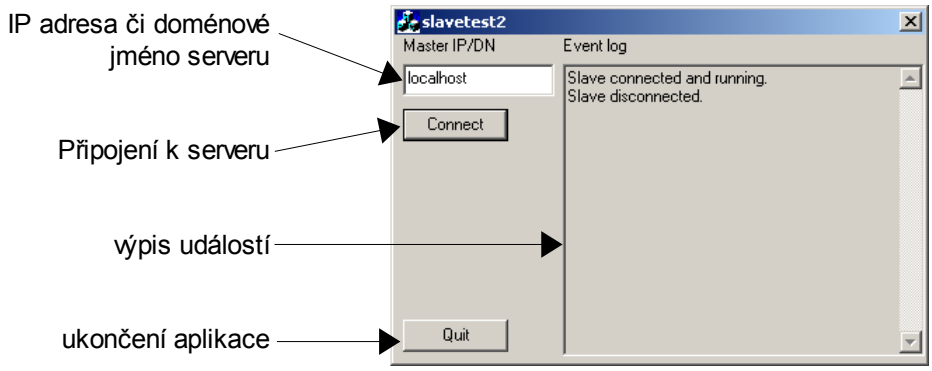
Obecně může libovolný otrok v clusteru obdržet úlohu rozpoznávní slova patřící libovolnému z klientů připojených k serveru. Ti mohou mít každý jinou konfiguraci rozpoznávače, v případě použití jediného rozpoznávače ve slave aplikaci by vznikal problém neustálého inicializování rozpoznávače (tvorba referencí řetězením hláskových modelů podle použitého slovníku a souboru s modely) s jinou konfigurací. V případě větších slovníků již může být čas strávený touto činností nezanedbatelný. Jednou zinicializované slovníky jsou proto dávány do úschovy (*cache*) a při příchodu nové úlohy se ověřuje, zda rozpoznávač s požadovanou konfiguraci již není v úschově. Je-li, provede se na něm rovnou rozpoznávání, pokud v úschově není, vytvoří se nový rozpoznávač s požadovanou konfigurací a použije se. Použitím cache pro slovníky sice může docházet k nárůstu paměti obsazené otrockou aplikací, paměťová náročnost vytvořených rozpoznávačů však není příliš velká, rozpoznávač se slovníkem o 10000 slovech obsadí přibližně 3MB paměti.

Pokud klient požaduje po otrokovi rozpoznání slova s použitím uživatelského slovníku (slova navíc k vybranému pevnému slovníku), je dočasně vytvořen slovník obsahující uživatelská slova, na kterém je realizováno přídavné rozpoznávání (navíc k rozpoznávání na pevném slovníku). Výsledky rozpoznávání z pevného a dočasného slovníku jsou poté vyhodnoceny a seřazeny běžným způsobem.

4.4.3.2 Výsledná aplikace

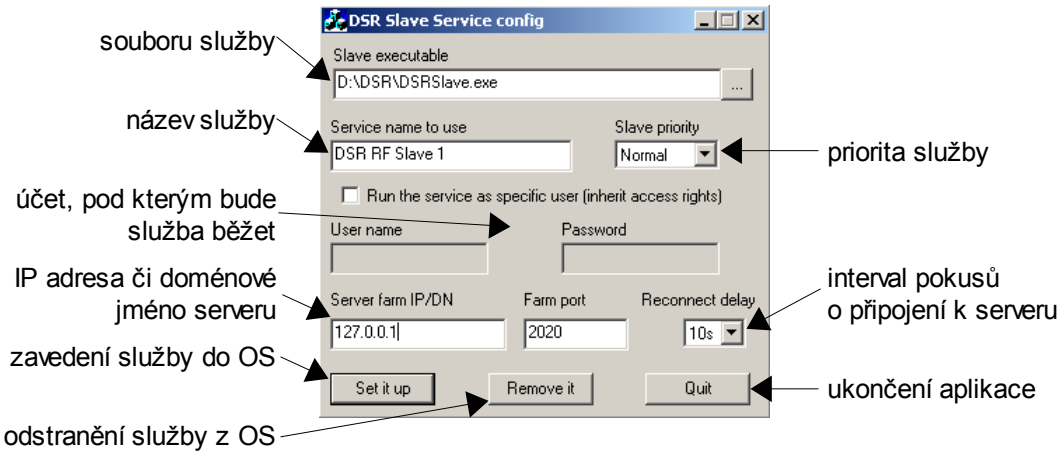
Stejně jako aplikace serveru byl i clusterový otrok vypracován ve dvou variantách – testovací s grafickým rozhraním (*slavetest2.exe*) a verze pro provoz jako služba OS (*DSRSlave.exe*). Ta se zavádí do systému vlastním konfiguračním programem (*RFSlaveSetup.exe*).

Rozhraní testovacího clusterového softwaru znázorňuje obr. 10. Stačí jen zadat IP adresu či doménové jméno serveru a připojit se, aplikace používá výchozí port 2020. Po úspěšném připojení již otrok pracuje na úlohách zadaných serverem až do ukončení aplikace či výpadku spojení.



obr. 10 – rozhraní testovací otrocké aplikace

Slave ve formě služby OS je zaváděn konfigurační aplikací, jejíž vzhled je na obr. 11, její činnost je analogická zaváděcí aplikaci určené pro server. Při instalaci otroka stačí uvést adresu serveru a umístění souboru služby, pokud není specifikován uživatelský účet, běží služba pod systémovým účtem. Slave aplikace v podobě služby pracuje autonomně, v případě ztráty spojení se zkouší znovu připojit k serveru v pravidelných intervalech (*Reconnect delay*). Stejně jako v případě serveru je možno využít testovací aplikaci k ověření připojení otroka k serveru, neboť služba nemá k dispozici běžné prostředky interakce s uživatelem. Veškeré události zapisuje do logovacího souboru (má stejný název a umístění jako soubor služby, odlišuje se jen příponou *log*).



obr. 11 – konfigurační aplikace otroka jako služby OS

5 Testy

5.1 Úspěšnost rozpoznávání

Testy úspěšnosti, se kterou navržený systém rozpoznává, nebyly realizovány ve smyslu běžných testů na velké testovací množině a měření rozpoznávacího skóre. Zejména proto, že vytvoření či zlepšování rozpoznávače nebylo předmětem této práce a jeho úspěšnost je považována za externí faktor. Realizovány byly pouze namátkové zkoušky s testovacím klientem, bylo spuštěno rozpoznávání v reálném čase a ověřováno správné rozpoznání vyřčených slov ze slovníku (ten obsahoval deset tisíc slov, jedná se o soubor *test10k.vcb*).

Při vyřčení asi sta slov byla relativní úspěšnost rozpoznávače kolem sedmdesáti procent (slovník nemá optimální skladbu slov, mnoho se jich liší jen v jedné hlásce, což způsobuje rozpoznávači problémy). Na menších slovnících (stovky slov, více se odlišujících) byla pozorována úspěšnost přes devadesát procent. Lze tedy prohlásit, že přepis rozpoznávače do objektové podoby následně použité v projektu proběhl korektně (nedošlo k jeho degradaci).

5.2 Zátěžové testy

Testy výkonnosti a škálování systému jsou primárním ukazatelem přínosu clusterového zpracování, na které je tato práce zaměřena. Byla pro ně vytvořena speciální testovací konzolová aplikace (*DSRBench*), jejíž parametry jsou zadávány z příkazové řádky pro snažší použití v dávkových souborech (.bat).

Aplikace vytváří parametrem zadaný počet klientských objektů, které se připojí k serveru. Paralelní zátěž serveru je simulována cyklickým zasíláním vybraného audio souboru (určen též parametrem, obsahující realizaci jednoho slova) skrz všechny připojené klienty. Měřeny jsou následující charakteristiky – ping (prodleva v komunikaci server-klient), hrubý výkon systému (za jak dlouho rozpozná server určité množství slov) a průměrná + maximální prodleva rozpoznávní slova (mezi dokončením slova a obdržením výsledků rozpoznávání) v závislosti na počtu paralelně pracujících klientů (ten probíhá vždy interval 1 až počet připojených klientů). Report z testu je přidáván do vybraného souboru.

Bohužel, měření času ve Windows je silně závislé na posloupnosti, ve které přepíná OS mezi vlákny. Tento fenomén se uplatňuje zejména při velkém množství aktivních vláken, v benchmarku má každý klientský objekt vlastní vlákno, což při jejich velkém počtu značně zkresluje výsledky měření času - čas nemůže být sejmut ihned při obdržení odpovědi ze serveru, ale až v momentu, kdy dostane příslušné vlákno přiděleno procesor. Výsledky pak mají tendenci být „schodovité“, zkreslené a vykazovat velký rozptyl, lze je proto považovat spíše za orientační.

5.2.1 Konfigurace testů

Testy byly zrealizovány v počítačové učebně B1, budova A, TUL. všechny počítače mají stejnou hardwarovou konfiguraci a jsou propojeny rychlým Ethernetem:

- CPU Pentium 4 (650) @ 3,4 GHz, jádro Prescott
- základní deska Intel D945PSN, čipset i945
- 2x1024 MB DDR2 RAM, taktovací frekvence 266MHz

Na jednom počítači byl spuštěn server, na dalším benchmark a na zbylých otrocké aplikace. Vždy po připojení dalšího otroka byl třikrát zrealizován test, výsledky poté zprůměrovány (díky výše uvedeným problémům při měření času).

Počet testovacích klientů byl třicet, celkový počet otroků osm. Při maximálním počtu připojených otroků byl ještě realizován test se sto připojenými klienty (extrémní zátěž). Testy probíhaly při výchozí konfiguraci rozpoznávače (*default* soubory), výchozí soubor slovníku obsahoval deset tisíc slov (*slov10k.vcb*). Jako audio soubor byl použit „*babicka2.wav*“ (jak název napovídá, jde o promluvu slova babička).

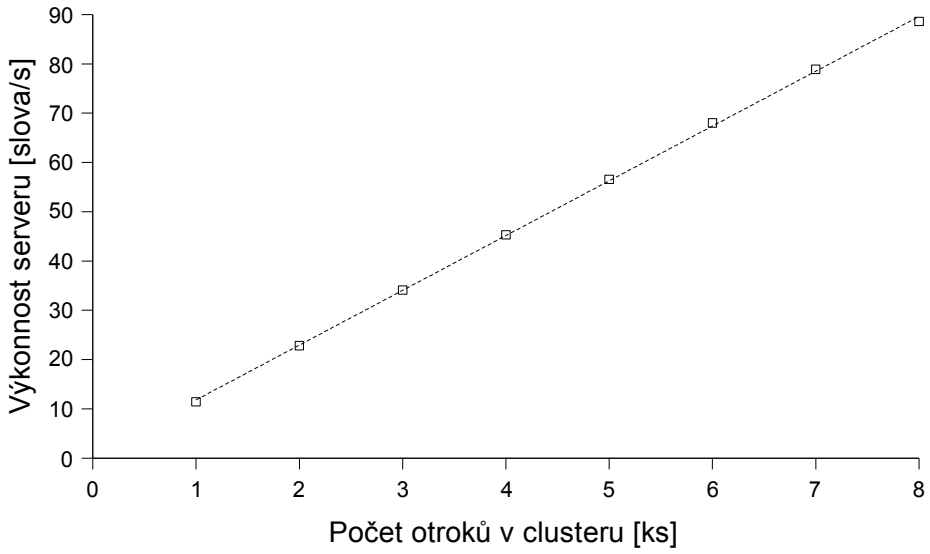
5.2.2 Výsledky testů a hodnocení

Vzhledem k testování na rychlé síti byly výsledky ping-testů hluboko pod 1 ms, lokální síť nezpůsobuje znatelnou prodlevu.

Výsledky testů hrubé síly zobrazuje tab. 1, resp. graf 1. Lze z nich jednoznačně vyčíst lineární zavislost celkového výkonu na počtu clusterových počítačů. Celkový výkon při více připojených otrocích je takřka přesným násobkem výkonu clusteru s jedním otrokem, z čehož lze usuzovat na minimální arbitrážní náklady celého clusteru.

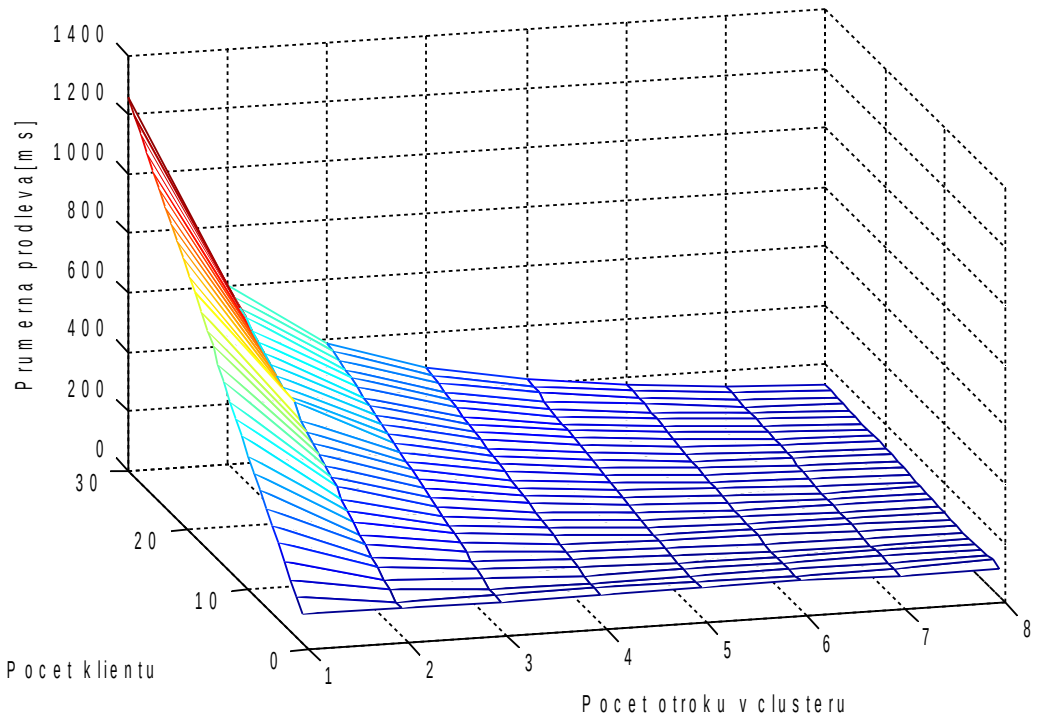
tab. 1 – rozpoznávací výkon serveru v závislosti na počtu otroků v clusteru

Počet otroků v clusteru [ks]	1	2	3	4	5	6	7	8
Výkon serveru [rozp. slova/s]	11,4	22,8	34,1	45,3	56,6	68,0	78,9	88,6

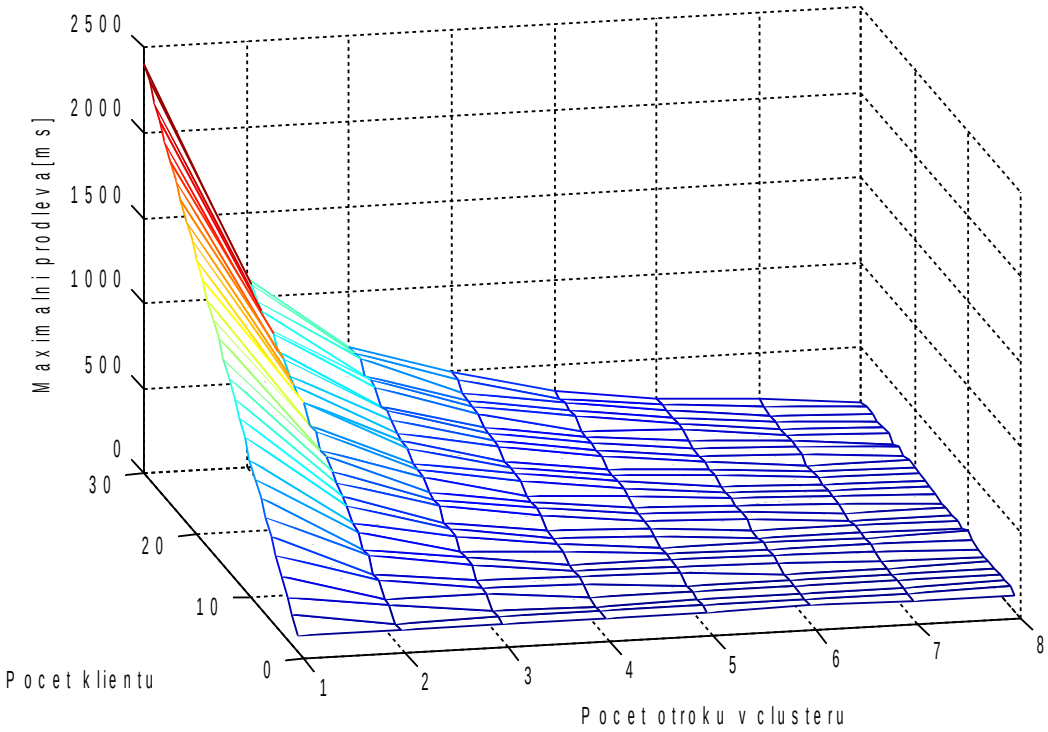


graf 1 – rozpoznávací výkonnost serveru v závislosti na počtu otroků

Výsledky testů zpoždění při rozpoznávání slova znázorňuje graf 2 a graf 3, odpovídající naměřené hodnoty jsou (vzhledem k jejich množství) na přiloženém CD v souboru „*benchmark100.txt*“. Dříve zmíněný graf představuje průměrnou prodlevu rozpoznávní slova při současném požadavku většího množství klientů. Při konstantním počtu otroků roste průměrná prodleva lineárně s množstvím současně obsluhovaných klientů, při konstantním počtu klientů pak vykazuje nepřímou úměru k počtu otroků v clusteru. Graf 3 zobrazuje extrémní (maximální) naměřené hodnoty prodlevy rozpoznávání (tj. prodlevy klienta, který musel ze všech připojených klientů čekat na výsledky rozpoznávání nejdéle). Minimální hodnota prodlevy je rovna času rozpoznávání jednoho slova otrokem, pro tyto testy se jedná o hodnotu kolem 100ms (pozorovatelná v grafu 4). Výsledky jsou v naprosté shodě s očekáváním, odpovídají naměřeným výkonnostním charakteristikám (graf 1).

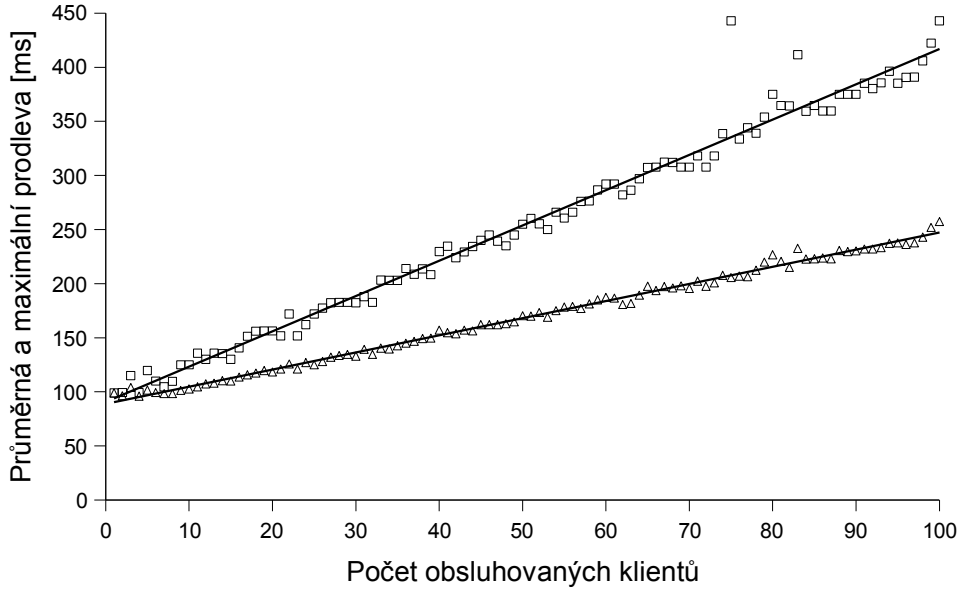


graf 2 – škálování systému, průměrná prodleva rozpoznání slova



graf 3 – škálování systému, maximální prodleva rozpoznání slova

Průběh zpoždění při masovém zatížení (až 100 klientů) ukazuje graf 3 (čtverce – maximální prodleva, trojúhelníky – průměrná prodleva rozpoznání slova), hodnoty čerpány též ze souboru „*benchmark100.txt*“. Je nutno upozornit, že všechny testy včetně tohoto představují nejhorší možnou variantu – slovo je pro všechny klienty detekováno a rozpoznáváno „najednou“ (v krátkém časovém rozmezí), při reálné operaci tato alternativa pravděpodobně nenastane.



graf 4 – závislost průměrné a maximální prodlevy na počtu klientů při 8 otrocích

Podle očekávání prodleva roste lineárně s počtem vznášených požadavků. Ukazuje se, že 8 otroků již přibližně postačuje na pokrytí stovky klientů, prodleva pro nejhorší případ kolem 400ms je pro člověka přijatelná.

K výsledkům testů je třeba dodat, že výpočetní náročnost rozpoznávání je silně závislá na rozpoznávaném slově (rozpoznávač používá akceleraci pomocí prořezávání hypotéz – tzv. pruningu), čili je nelze brát za absolutní míru výkonnosti. Také reálná prodleva při rozpoznávání slova může být navýšena vlivem paketizace klientem odesílaných příznakových vektorů (nejsou zasílány po jednom, ale ve větších blocích za účelem minimalizace síťové režie).

V průběhu testu se 100 klienty a 8 otroky bylo pozorováno vytížení procesoru serveru, to dosahovalo špičkově asi 30%. Lze tudíž předpokládat, že server zvládne řízení clusteru o 30 i více stejných počítačích (shodných se serverem), aniž by se stal limitujícím faktorem celkového výkonu.

6 Diskuse

6.1. Splnění cílů diplomové práce

Hlavní cíle práce byly splněny, byl vytvořen funkční systém distribuovaného rozpoznávání řeči (izolovaných slov a krátkých frází) s použitím clusterových výpočtů. Systém je stabilní i při extrémním vytížení (benchmark se 100 klienty), při dostatečném zázemí clusteru (větší množství běžných počítačů) poskytuje rozpoznávací potenciál těžko (či s neúměrně vyššími náklady) realizovatelný jediným počítačem.

Systém nevyžaduje rozsáhlou instalaci softwaru, přiložené konfigurační programy jsou přehledné. V případě výpadků počítačové sítě dochází automaticky k pokusům o obnovení spojení v rámci clusteru. Aplikace realizující cluster nevyžadují přihlášení uživatele k počítači, spouští se automaticky jako služba při startu operačního systému.

6.2 Přínos, možnost uplatnění

Práce ověřila vhodnost použití clusterových výpočtů pro potřeby DSR systémů a nastínila tak možnost výrazné úspory při jejich realizaci. Dále byla vyřešena problematika výpočetních clusterů v obecné rovině, může tak sloužit jako východisko při návrhu jiných systémů vyžadujících extrémní výpočetní výkony a krátkou dobu odezvy.

Samotné distribuované rozpoznávání řeči opřené o dostatečný výpočetní výkon clusteru lze uplatnit např. v centralizovaných hlasových systémech či může poskytnout zázemí pro hlasové ovládání velkého množství jednoduchých zařízení (spotřební elektronika, elektrotechnika apod.). Potřebné propojení do sítě v rámci objektu pak může být realizováno buď zvláštním rozvodem, bezdrátově či modulátory elektrorozvodné sítě, na větší vzdálenosti pak za pomoci standardního internetového připojení. Nespornou výhodou DSR je především možnost centrální správy (updatu).

6.3 Nevýhody použitého řešení

Za největší nevýhodu celého projetku lze označit jeho složitost a velké množství elementů, kdy selhání některého z kritických prvků (např. serveru či spojení

server-klient) má za následek nefunkčnost (alespoň z pohledu klienta). Rozložení procesu rozpoznávání na více počítačů též způsobuje horší diagnostikovatelnost při výskytu chyby. V tomto ohledu má rozpoznávač umístěný přímo do zařízení nesporné výhody – v případě výskytu chyby se její hledání omezuje jen na dané zařízení.

Obecnou nevýhodou DSR je nutnost dodatečného připojení zařízení do datové sítě, pokud toto ještě neexistuje (např. u mobilních telefonů tento problém odpadá), a závislost na jeho stabilitě. V případě použití konvenčního řešení (Ethernet) navíc přibývá kabeláže.

6.4 Možnosti zlepšení

V průběhu realizace projektu se vyskytlo velké množství drobných problémů (dáno celkovou komplexností), nezbyl proto čas na dokončení jistých, původně zamýšlených prvků.

Server neobsahuje uživatelské účty a práva, veškerý přístup je anonymní (server jen zaznamenává IP adresy připojovaných klientů a otroků) – zavedením přihlašování lze zlepšit bezpečnost serveru, vymezit možnosti jednotlivých klientů (např. omezit možnost konfigurace rozpoznávače apod.).

Verze serveru pro ostrý provoz běží jako služba a neobsahuje žádné rozhraní pro interakci s uživatelem, např. pro účely diagnostiky chyb při náběhu či pro správu serveru za jeho chodu. Tento probém lze vyřešit naprogramováním speciální aplikace, která bude se službou spolupracovat prostředky meziprocesní komunikace OS.

Nároky na propustnost sítě a jednoho připojeného klienta jsou přibližně 5,5KB/s (44 Kbit/s), při velkém množství klientů dojde snadno k zahlcení datového spoje. DSR systémy proto integrují kompresní algoritmy pro rozpoznávací příznaky, nejčastěji používané jsou ztrátové komprese mající základ ve vektorové kvantizaci (např. Split Vector Quantization), které redukují datový tok na osminu až desetinu. Aplikace této technologie na klient-server komunikaci (posílání rozpoznávacích příznaků) by vedla k možnosti připojit více klientů při stejné propustnosti, za cenu zvýšení hardwarových nároků na klienta.

Závěr

Práce ukazuje, že přínosy clusterového zpracování lze s výhodou uplatnit i v počítačovém rozpoznávání řeči. Jako její součást vznikl obecný systém pro realizaci clusterových výpočtů, který byl posléze aplikován na problematiku DSR, je však použitelný pro clusterové zpracování libovolných úloh.

Výsledky testů překvapivě ukazují zanedbatelný vliv arbitráže a komunikace v clusteru, celkový výkon představuje takřka prostý součet výkonů clusterových počítačů.

Do budoucna lze předpokládat rozmach hlasového ovládání různých zařízení ve snaze o zvýšení komfortu při jejich obsluze. Nasazení distribuovaného rozpoznávání řeči umožňuje redukovat náklady na konstrukci těchto zařízení, použití clusterového zpracování může výrazným způsobem zlevnit systémy DSR, které budou čelit rostoucím nárokům na výkon.

Seznam použité a odkazované literatury

[1] PELC, M. *Návrh masivního řečového rozpoznávacího stroje s distribuovanými výpočty*. Liberec, 2005. Ročníkový projekt na fakultě mechatroniky a mezioborových inženýrských studií TUL. Vedoucí ročníkového projektu Miroslav Holada.

[2] NOUZA, J. (ed.): *Počítačové zpracování řeči. Sborník prací*. TUL 2001

[3] ŽĎÁNSKÝ, J. – NOUZA, J.: *Detection of Acoustic Change-Points in Audio Records via Global BIC Maximization and Dynamic Programming*. In: Interspeech 2005, September, 2005, Lisboa, Portugal, s. 669-672. ISSN 1018-4074

[4] HUANG, X. – ACERO, A. – HON, H-W. *Spoken language Processing, A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, New Jersey, 2001. ISBN 0-13-022616-5

[5] PSUTKA, J.: *Komunikace s počítačem mluvenou řečí*. Academia, Praha, 1995. ISBN 80-200-0203-0

[6] EKSLER, V. – POLÁČEK, V. *Internet speech recognizer for blind separated signal evaluation in telecommunications and signal processing*. Telecommunications and Signal Processing TSP-2005. Brno, 2005, s. 74 - 77, ISBN 80-214-2972-0

[7] HOLADA, M. – PELC, M.: *Distributed Speech Recognition System Using Parallel Processing*. In: Electronic Speech Signal Processing 2005, Semtember 2005, Prague, Czech Republic, s. 273-276. ISBN 3-938863-17-X

[8] The Microsoft Developer Network (MSDN) Online, Platform SDK.
Windows User Interface (Messages and Message Queues)
Windows Base Services (Processes and Threads)
Windows Sockets / GQOS
URL: <http://msdn.microsoft.com>

Přílohy

A – schema činnosti výsledného systému

CD – obsah CD je uveden v jeho kořenovém adresáři v souboru *readme.txt*