České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Vzdálené rozvrhování úloh s využitím
Scheduling toolboxu pro Matlab

Praha, 2007 Jan Martinský

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky *Školní rok:* 2005/2006

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Jan Martinský

Obor: Technická kybernetika

Název tématu: Vzdálené rozvrhování úloh s využitím Scheduling toolboxu pro Matlab

Zásady pro vypracování:

1. Seznamte se se základy teorie grafů a teorie rozvrhování. Prostudujte základní příkazy a možnosti Scheduling toolboxu pro Matlab.
2. Navrhněte a implementujte vhodné webové uživatelské rozhraní pro vzdálenou práci s toolboxem. Pro komunikaci s toolboxem použijte Matlab Web Server.
3. Doplňte scheduling toolbox o vhodné algoritmy, jenž bude možné použít pro demonstraci možností toolboxu, při vzdáleném přístupu.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Michal Kutil

Termín zadání diplomové práce: zimní semestr 2005/2006

Termín odevzdání diplomové práce: leden 2007

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

v zast.

prof. Ing. Vladimír Kučera, DrSc.
děkan

V Praze dne 20.01.2006

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 25.5.2007 .....

.....
podpis

Poděkování

Na úvod práce bych rád poděkoval ing. Michalu Kutilovi, který měl se mnou trpělivost a byl vždy ochoten mi poradit a bez jeho rad bych nebyl schopen práci dokončit. Dále bych rád poděkoval rodině a kamarádům, za morální podporu.

Abstrakt

Tato práce se zabývá vzdáleným rozvrhováním s využitím TORSCHE Scheduling Toolboxu pro Matlab. Vzdálené rozvrhování je zprostředkováno webovým rozhranním implementovým v jazycích PHP a JavaScript. Pro pohodlnější a jednodušší obsluhu je použito konceptu AJAX. Přístup uživatelů k rozhranní je založen na systému uživatelských účtů. Pro uchovávání dat je použita relační databáze MySQL. Úlohy zadávané uživateli prostřednictvím rozhranní jsou aplikací nezávisle řešeny na Matlab serverech. Vzhledem k časově náročnějším výpočtů rozvrhovacích úloh je použito více Matlab serverů, které řeší různé úlohy paralelně. Pro ošetření možných kolizí paralelního zpracování je navržen systém zámků. Spouštění Matlab serverů je zajištěno prostřednictvím CGI skriptu. Komunikaci Matlab serverů s databází probíhá prostřednictvím SOAP serveru, který poskytuje Matlab severům webové služby. Na závěr jsem implementoval rozvrhovací algoritmy Hu a Coffman and Graham.

Abstract

This thesis deals with the remote scheduling using TORSCHE Scheduling Toolbox for Matlab. The remote scheduling process is mediated by the web interface, that was implemented using PHP and JavaScript programming languages and AJAX. User access is based on user accounts system, that allows creating unique user accounts. Data is stored in relation database MySQL. The tasks created by users using web interface are solved on Matlab servers. Considering to time consuming computations, more Matlab servers are used in aplication to solve scheduling problems independently. The system of locks was implemented for solving possible conflicts of Matlab servers while accessing database. The Matlab servers are initialised throught CGI script. Comunication between Matlab server and database is mediated by SOAP server, that offers web services to Matlab servers. Two scheduling algorithms Hu`s algorithm and Coffman and Graham algorithm were implemented.

Obsah

Seznam obrázků

Seznam tabulek

Seznam kódů

1. Úvod

V dnešní době velkého rozvoje sítě Internet se používání webových aplikací stává běžnou součástí našeho života. Existuje nepřeberné množství webových aplikací, které nám usnadňují práci. Mezi nejpoužívanější patří například rozhranní pro přístup k e-mailové schránce, internetové obchody, diskusní fóra a další. Jsou nezávislé na platformě a pouhým vlastnictvím webového prohlížeče nám umožňují využití vzdálených nástrojů bez zatěžování našeho systému. Interaktivní webová rozhranní navržená za použití moderních technologií se obsluhou a použitím blíží používaným desktopovým aplikacím a nacházejí si proto čím dál tím větší oblibu. Pro vytváření interaktivních webových aplikací se využívá technologií umožňující skriptování na straně klienta. K technologiím umožňující skriptování na straně klienta patří například JavaScript, JScript, VBScript a další. V dnešní době se rychle ujalo využívání konceptu AJAX, který kombinuje technologie JavaScript a XML a umožňuje výměnu dat mezi klientem a serverem bez nutnosti načítání celé webové stránky. Mezi nevýhody použití skriptování na straně klienta je nutnost použití webového prohlížeče, který daný skript podporuje. Z tohoto důvodu je nutné vytvářet aplikace tak, aby se daly použít i bez skriptování na straně klienta.

Cílem této práce je vytvořit interaktivní webové rozhranní, jednoduché a srozumitelné pro uživatele, umožňující řešení rozvrhovacích problémů a navrhnout vhodné rozvrhovací algoritmy. Rozvrhování tvoří nedílnou součást jak chodu firem a podniků, tak i našeho života. Efektivně rozvržená práce a použití výrobních prostředků šetří čas i peníze. Na katedře řídící techniky je proto vyvíjen nástroj pro řešení rozvrhovacích problémů TORSCHE Scheduling Toolbox pro Matlab[1]. Navrhované webové rozhranní má umožnit rozvrhování s využitím tohoto nástroje. Vzhledem k tomu, že vytvoření rozvrhu velké sady úloh je často časově a výpočetně velmi náročná úloha, je využití vzdáleného nástroje velmi užitečné. Uživatel pouze zadá úlohy, které potřebuje rozvrhnou a po nějakém čase si vyzvedne již hotový rozvrh.

Text této práce je rozdělen na šedm částí. Nejdříve je uveden stručný popis použitých technologií. V druhé části je uveden úvod do problematiky rozvrhování a popis použitých nástrojů souvisejících s rozvrhováním (TORSCHE, PsGantt). Třetí část obsahuje popis vytvořené aplikace pro vzdálené rozvrhování. Zde je uveden popis jednotlivých částí, ze kterých se aplikace skládá a jejich vzájemné propojení. Následující část popisuje implementované rozvrhovací algoritmy. Pátá část se zabývá testy vytvořené aplikace. V předposlední části je demonstrováno použití webového rozhranní a v závěru je uvedeno shrnutí dosažených výsledků.

[1] Dále jen TORSCHE

2. Použité technologie

2.1. XML

XML (eXtensive Markup Language) je obecný značkovací jazyk pro práci se strukturovanými daty, který byl vyvinut a standardizován konsorciem W3C. Tento jazyk je určen především pro výměnu dat mezi aplikacemi. V této aplikaci je XML využito pro uchování konfiguračích dat a komunikaci mezi některými částmi aplikace [XML].

2.2. HTML, XHTML

HTML (HyperText Markup Language) je značkovací jazyk pro hypertext založený na SGML (Standard Generalized Markup Language). Tento jazyk byl standardizován kosorciem W3C a není závislý na platformě. Používá se pro vytváření stránek pro WWW (World Wide Web). Pomocí standardizovaných návěstí (tagů) definuje význam strukturovanému textu [HTML].

XHTML (eXtensive HTML) je značkovací jazyk pro tvorbu hypertextových dokumentů, který byl vyvinut konsorciem W3C, který aplikuje technologii XML [XHTML].

2.3. CSS

CSS (Cascade Style Sheets) je jazyk pro popis zobrazení dokumentu napsaného pomocí značkovacího jazyka. Nejčastěji se používá pro popis zobrazení webových stránek napsaných pomocí HTML a XHTML. Specifikace jazyka je udržována konsorciem W3C [CSS].

2.4. PHP

PHP je skriptovací jazyk nezávislý na platformě vyvinutý pro dynamické generování webových stránek. V roce 1994 ho vytvořil Rasmus Lordorf. Používá se pro server-side scripting, kdy skript je vykonáván na straně serveru a k uživateli je přenášen jen výsledek jeho činnosti [PHP].

2.4.1. Session

Session udržují informace o stavu PHP aplikace a práci uživatele pomocí superglobální proměnné $_SESSION. Pomocí session můžeme jednoznačně identifikovat uživatele a dle těchto informací mu umožnit práci s aplikací.

2.5. JavaScript

Multiplatformní jazyk založený na konceptu prototype-based programování. Tedy objektově orientované programování bez používání tříd s využíváním klonů již existujících objektů. Běžně se JavaScript používá ve Webových aplikacích pro client-side scripting, kdy skript je vykonáván na straně uživatele bez účasti serveru [JavaScript].

2.5.1. AJAX

AJAX (Asynchronous JavaScript And XML) slouží pro vývoj interaktivních Webových aplikací. Umožňuje výměnu pouze nezbytných dat mezi aplikací a serverem, takže není potřeba nahrávat celou Webovou stránku [AJAX].

2.6.SQL

SQL (Structured Query Language) je dotazovací jazyk používaný pro práci s relační databází standardizován standardizačnímy instituty ANSI a ISO. Jedná se o deklarativní programovací jazyk, který popisuje co se má udělat, nikoli jak se to má udělat [SQL].

2.7. SOAP

SOAP (Simple Object Access Protocol, Service Oriented Architecture Protocol) je protokol pro výměnu dat prostřednictvím počítačové sítě ve formátu XML. Je složen z obálky, kódovacích pravidel a konvence pro RPC [2][SOAP].

2.8. Web services

Web services je standardizovaný popis komunikace aplikací po síti. Jedná se o komunikaci ve formě XML zpráv pomocí protokolu SOAP. Komunikace je nezávislá na použitých platformách i programovacích jazycích aplikací [Web services].

2.9. WSDL

WSDL (WebService Description Language) popisuje komunikaci s web services ve formátu XML. Určuje jaké funkce web services nabízí, jaké mají parametry a jaké jsou návratové hodnoty.

2.10. CGI

CGI (Common Gateway Interface) je standard, který definuje jakým způsobem spolu komunikují aplikační server a externí skript. Externí skript může být napsán v jakémkoli programovacím jazyce. CGI se běžně používá pro dynamické generování webových stránek.

[2] RPC (Remote Procedure Call) je technologie umožňující vzdálené volání metod prostřednictvím XML zpráv.

3. Rozvrhování

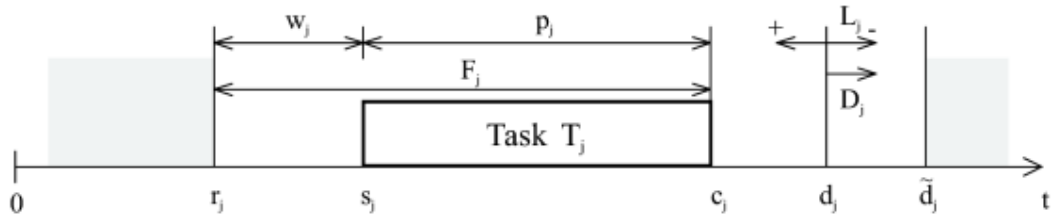
Tato práce se zabývá vzdáleným rozvrhováním, proto je v této kapitole uveden stručný úvod do problematiky rozvrhování.

3.1. Základní pojmy

Rozvrhování je přiřazení zdrojů jednotlivým úlohám v čase. Úlohy jsou vykonávány na procesorech. Vstupem do rozvrhovacího procesu je množina úloh, zdrojů, omezení a penalizačních funkcí. Výsledkem je nový rozvrh s pořadím vykonávání jednotlivých úloh a jejich přiřazení na procesory. Rovrhování má velkou škálu vyžití od informatiky, administrativy, dopravy po stavebnictví. Podrobnější popis naleznete v literatuře [Hanzálek].

3.1.1. Úloha

Základním prvkem rozvrhování je *úloha (task)*. Úloha představuje jeden proces nebo událost v čase. Pro popis se používají parametry uvedené níže.



Obr. 3.1: popis úlohy

- p_j ***processing time (doba vykonávání)***. Doba, která je potřeba pro vykonání úlohy na procesoru bez přerušení
- r_j ***release time (okamžik disponibility)*** . Čas , kdy je úloha připravena k vykonání.
- d_j ***deadline (poslední přípustný okamžik dokončení)***. Čas do kdy musí být úloha dokončena, aby byla povžovaná za úspěšně vykonanou.
- $\tilde{d}_j$ ***duedate (okamžik požadovaného doklončení)***. Čas do kdy by úloha měla být dokončena, jinak dojde k penalizaci kritéria.
- w_j ***waiting time (prodleva)***. Čas po který úloha připravená k vykonávání čeká na přiřazení procesoru.
- ***sj*** ***starting time (okamžik započetí)***. Okamžik, kdy je započato vykonávání úlohy.
- c_j ***completion time (okamžik dokončení)***. Okamžik, kdy je dokončeno vykonávání úlohy.
- F_j ***flow time/response time (reakční doba)***. $F_j = c_j - r_j$
- L_j ***lateness (zpoždění)***. Zpoždění dokončení úlohy vzhledem k okamžiku požadovaného dokončení $L_j = c_j - \tilde{d}_j$.
- D_j ***tardliness (zpoždění)***. Zpoždění pouze v záporném smyslu $D_j = \max\{c_j - \tilde{d}_j, 0\}$.

3.1.2. Standardní notace

Pro popis rozvrhovacího problému byl zaveden jednotný popis pomocí notace $\alpha \mid \beta \mid \gamma$, kterou definovali Graham a Błażewicz.

α – definuje počet a typ procesorů

- **univerzální** – na procesoru může být vykonána libovolná úloha
  - **P** identické procesory – stejná rychlost vykonávání úlohy na všech procesorech
  - **Q** uniformní procesory – doba vykonání úlohy je nepřímo úměrná jeho rychlosti
  - **R** rozdílné – doba vykonání úlohy je na každém procesoru jiná
- **specializované** – procesor může vykonávat pouze určité úlohy
  - **O** open shop – nezávislé úlohy bez relací následnosti
  - **J** job shop – úlohy jsou vázány relacemi následnosti, které nejsou totožné pro všechny joby
  - **F** flow shop – úlohy v jobech jsou vykonány ve stejném pořadí

β – určuje omezení problému

- **prec** prcedenční omezení, kdy okamžik započetí nějaké úlohy je závislý na dokončení jiné úlohy
- **pmtn** preempce, úloha může být přerušena a dokončena později
- **res** přídavné zdroje
- $\mathbf{r_j, p_j, d_j, d_j^{\sim}}$ omezení okamžiku disponibility, doby vykonávání, okamžiku požadovaného dokončení, posledního přípustného okamžiku dokončení
- **no-wait** úlohy nemohou být ukládány do meziskladů
- **chain** řetězová precedenční závislost
- **tree** stromová struktura precedenční závislosti

γ – značí kritérium optimalizace

- $\mathbf{C_{max}}$ minimalizace celkové délky rozvrhu
- $\Sigma \mathbf{C_j}$ minimalizace součtu okamžiků dokončení všech úloh
- $\Sigma \mathbf{w_j C_j}$ minimalizace váženého součtu okamžiků dokončení
- $\mathbf{L_{max}}$ minimalizace zpoždění

3.1.3. Složitost úloh

Pro algoritmy, které řeší nějakou úlohu lze určit jejich výpočetní složitost. Řekněme, že úloha *U* má horní odhad složitosti *f(n)*, jestliže existuje algoritmus, který řeší úlohu *U* a má časovou složitost *O(f(n))*. Jinými slovy, horní odhad složitosti úlohy nám dává jakýkoli algoritmus, který danou úlohu řeší a jehož časovou složitost umíme odhadnout shora [Mařík].

Úlohy můžeme podle složitosti rozdělit do tříd:

P Třída všech úloh, které lze řešit polynomiálním algoritmem se složitostí *O(p(n))*.

NP Třída úloh, pro které existuje nedeterministický algoritmus pracující v polynomiálním čase [Stibor].

NP-úplné Úloha U je NP-úplná, jestliže je NP úlohou a každá NP úloha se na se na U polynomiálně redukuje [Stibor].

NP-těžké Problém A je NP-těžký, pokud každý problém z NP lze redukovat na problém A [Hanzálek].

3.2. TORSCHE Scheduling toolbox for Matlab

TORSCHE(Time Optimisation of Resources, SCHEduling) je toolbox pro výpočetní prostředí Matlab obsahující algoritmy pro řešení úloh rozvrhování. Jedná se o nástroj šířený pod licencí GNU General Public Licence, který je vyvíjen na Fakultě elektrotechnické ČVUT. Více informaci o TORSCHE naleznete na www [TORSCHE].

3.2.1. Vytváření úloh

Způsob zadávání rozvrhovacího problému ke zpracování pomocí scheduling toolboxu je možné najít na WWW [TORSCHE]. Zde uvedu pouze stručný úvod do syntaxe vytváření základních objektů a funkcí použitých v toolboxu.

Pro vytvoření objektu úlohy slouží příkaz *task* následovaný parametry úlohy a parametry *weight* a *processor*. *Weight* určuje prioritu úlohy vzhledem k ostatním úlohám. *Processor* definuje číslo procesoru, na kterém se musí úloha vykonávat. Ukázka syntaxe pro vytvoření objektu `task`:

```
>> uloha = task('uloha',4,0,7,6,1,2)

Task "uloha"

 Processing time: 4

 Release time:    0

 Deadline:        7

 Due date:        6

 Processor:       2
```

Úlohy vytvořené příkazem `task` můžeme příkazem `taskset` spojit do sad úloh. Pro vytvoření objektu `taskset` použijeme syntaxi:

```
>> set = taskset([uloha uloha uloha]);
```

Pro vyjádření precedenčních omezení v sadě úloh slouží také příkaz `taskset`, kde precedenční omezení jsou zadána jako druhý vstupní parametr ve formě čtvercové matice `prec`. Matice precedenčních omezení je rozměru $i \times j$, kde i je počet řádků a j je počet sloupců a platí $i = j = n$, kde n značí počet úloh. Potom pokud úloha i má být vykonána před úlohou j `prec(i,j)=1`, jinak `prec(i,j)=0`. Ukázka definování precedenčních omezení v sadě úloh:

```
>> set = taskset(set,[0 1 0;0 0 0;0 1 0]);
```

Pro definování rozvrhovacího problému slouží příkaz `problem`, kde vstupním parametrem je problém zapsaný standardní notací. Objekt typu problem vytvoříme:

```
>> p = problem('P2|prec,pj=1|Cmax');
```

Takto vytvořenou sadu úloh můžeme rozvrhnout pomocí algoritmů TORSCHE. Vstupní parametry algoritmu jsou sada úloh, rozvrhovací problém a další parametry vyžadované pro vstup dané funkce. Pro zadání úlohy pro algoritmus Coffman and Graham použijeme zápis:

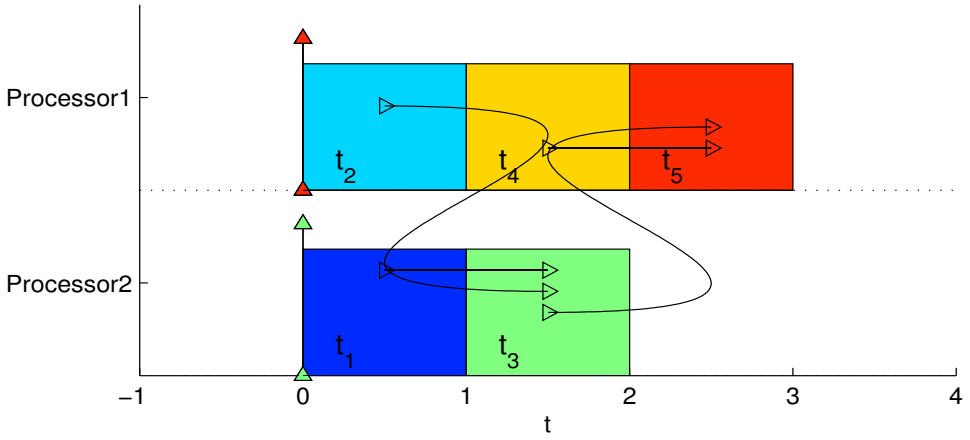
```
>> set = taskset([1 1 1 1 1]);

>> set = taskset(set,[0 0 1 0 0;0 0 1 0 0;0 0 0 0 1;0 0 0 0 1;0 0 0 0 0]);

>> schedule = coffmangraham(set,p);
```

Výsledné rozvrhy, které vzniknou aplikací rozvrhovacích algoritmů můžeme zobrazit ve formě ganttových diagramů pomocí funkce `plot`:

```
>>plot(schedule);
```



Obr. 3.2: TORCHE ganttův diagram

3.3. PsGantt

Pro zobrazení výsledků rozvrhovacího procesu se běžně používá ganttových diagramů, které umožňují názorný přehled výsledného rozvrhu. PsGantt je nástroj pro kreslení ganttových diagramů vytvořený Antonínem Karolíkem [Karolík] a zpřístupněn pomocí web services Radovanem Černým [Černý]. Tento nástroj umožňuje prostřednictvím web services vytvořit ganttův diagram pro daný rozvrh v různých grafických formátech.

4. Aplikace

Úkolem aplikace je umožnit prostřednictvím webového rozhranní vzdálené rozvrhování. K rozvrhování má být použito TORSCHE.

4.1. Požadavky

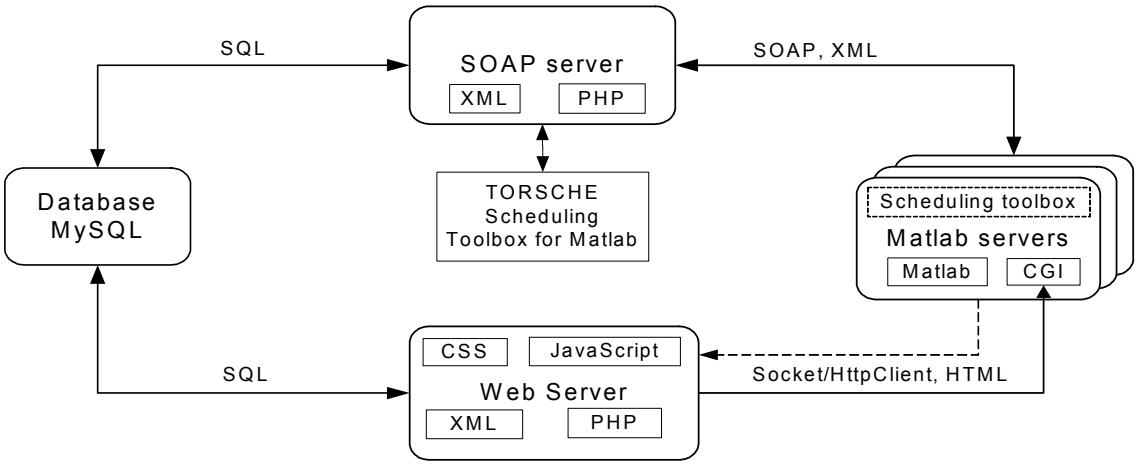
Rozborem zadání a konzultací s vedoucím práce byly stanoveny hlavní požadavky na aplikaci, které jsou pro správnou funkci nezbytné. Mezi hlavní poždavky patří:

- ***Obecnost*** - Důležitým požadavkem je, aby existoval obecný postup zadávání a zobrazení úloh přes webové rozhranní nezávislý na zvoleném algoritmu z TORSCHE.
- ***Rozšiřitelnost*** - Je nezbytné, aby bylo možno aplikaci jednoduše rozšířit o nové algoritmy TORSCHE bez výrazných změn v aplikaci.
- ***Jednoduchost*** - Pro snadnou práci uživatele s aplikací je potřeba jednoduché a intuitivní ovládání, které umožní přehledný způsob zobrazení výsledných rozvrhů.
- ***Přenositelnost*** - Co nejlepší přenositelnolnost v rámci různých platforem.
- ***Rychlost*** - Vzhledem k časově náročným výpočtům je nutné zajistit rychlou odezvu.

4.2. Koncepce aplikace

Koncepce aplikace vychází z potřeby vytvořit webové rozhranní a vytvořit jeho propojení s TORSCHE při splnění uvedených požadavků. Cílem je, aby uživatelem vytvořená úloha pomocí webového rozhranní byla odeslána na Matlab server k vyřešení a rozhranní zpět obdrželo výsledný rozvrh. Nejprve je nutné vytvořit webové rozhranní a Matlab server s TORSCHE sloužící pro výpočty. Webové rozhranní je implementováno pomocí technologie PHP, pro snadnější práci s ním je využito technologie JavaScript a konceptu AJAX. Matlab server je tvořen výpočetním prostředím Matlab a TORSCHE Scheduling Toolboxem pro Matlab. Protože se předpokládá větší časová náročnost řešení rozvrhovacích algoritmů je samotný výpočet oddělen od práce s rozhraním a vykonáván nezávisle na Matlab serveru. K tomu je nezbytné vytvořené úlohy a jejich řešení uchovávat v databázi. Výměna dat mezi web serverem a Matlab serverem je tedy složena z komunikace web server s databází a Matlab server s databází. Pro Matlab server je nutné navrhnout způsob komunikace s databází. K tomuto účelu byla vytvořena aplikace serveru poskytující pomocí web services služby pro výměnu dat s databází. Pro samotné spuštění výpočtů na Matlab serveru je vytvořen CGI skript spustitelný z web serveru. Klíčové je zajištění rychlé odezvy na zadání úlohy. K tomuto účelu je vytvořen způsob paralelního zpracování, kdy je pro výpočty použito více Matlab serverů paralelně řešících zadané úlohy. Aplikace je vytvořena za použití běžně dostupných technologií a standardů čímž je zajištěna dobrá přenositelnost.

4.2.1. Struktura

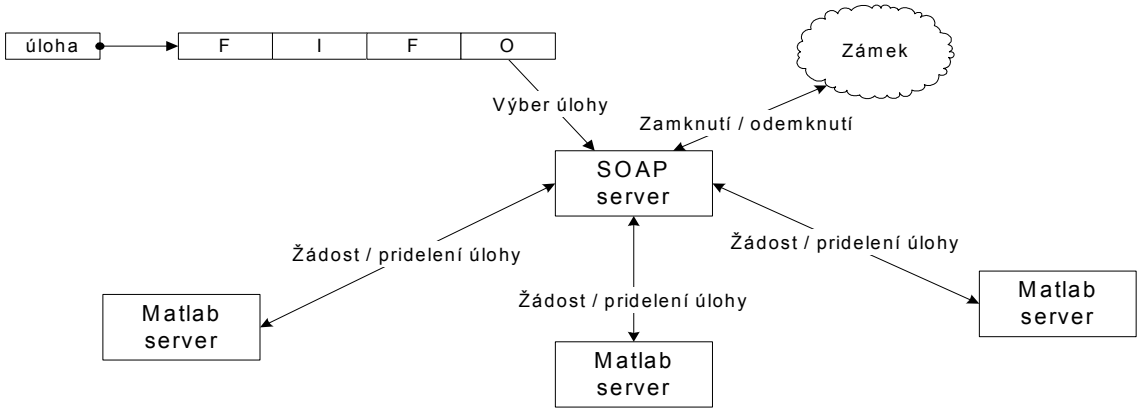


Obr. 4.1: struktura aplikace

Aplikace je složena ze čtyř částečně nezávislých částí, které zastávají různé funkce. První částí je webový server s rozhranním implementovaným pomocí technologie PHP, které je určeno pro komunikaci uživatele s aplikací. Rozhranní ověřuje identitu uživatele, ukládá zadání úloh do databáze, čte řešení z databáze a posílá Matlab serveru informaci o zadání úlohy. Druhou částí je relační databáze MySQL, kde se uchovávají data potřebná pro řešení úloh, vyřešené úlohy, uživatelská data a údaje o stavu systému. Zároveň je databáze využita pro systém zámků. Třetí částí je Matlab server, na kterém jsou zadané úlohy vyřešeny pomocí TORSCHE. Komunikace Matlab serveru a databáze je zprostředkována pomocí SOAP serveru, který z dat zadaných k vyřešení generuje kód pro Matlab server a vrácené výsledky ukládá do databáze. SOAP server také umožňuje update skriptů potřebných pro výpočet, například TORSCHE.

4.2.2. Paralelní zpracování

Nespornou výhodou webové aplikace je možnost nezávislé práce více uživatelů ve stejný okamžik, kteří mohou do systému zadávat své úlohy k vyřešení. Tato žádoucí vlastnost však s sebou přináší negativní dopad ve formě zvýšených nároků na výpočetní kapacitu serveru. Pokud výpočetní kapacita nedostačuje je úloha zařazena do fronty a dochází k prodloužení doby, kterou aplikace potřebuje na její vyřešení. Je proto nezbytné zajistit, aby se tento dopad promítl co nejméně do práce uživatelů s rozhranním. Toho je docíleno paralelním zpracováním úloh. Při paralelním zpracování se využívá více Matlab serverů, které provádějí výpočty paralelně a nezávisle na sobě. Úlohy zadané uživatelem k vyřešení jsou uloženy do databáze, kde jsou řazeny do fronty typu FIFO. Pokud je některý Matlab server připraven k řešení, zažádá SOAP server o přidělení úlohy. Tímto způsobem je využíváno maximální kapacity serveru pro řešení rozvrhovací úlohy. Pokud je potřeba řešit více úloh ve stejný okamžik je použitím paralelních výpočtů úloha vyřešena dříve.



Obr. 4.2: paralelní zpracování

Klíčovým prvkem paralelního zpracování je bezpečný způsob přiřazování úloh na jednotlivé Matlab servery k vyřešení. K tomuto účelu slouží SOAP server, který zprostředkovává komunikaci Matlab serverů s databází pomocí web services. Bezpečné přiřazení úloh zajišťuje pomocí systému zámků[3].

4.3. Databáze

V aplikaci se pracuje s velkým množstvím dat, která jsou uchovávána v relační databázi, která umožňuje pohodlný a rychlý přístup k uloženým datům pomocí jazyka SQL. Použití databáze je vhodné také z toho důvodu, že ji lze použít pro realizaci systému zámků pro paralelní zpracování úloh.

4.3.1. Použitá databáze

Pro realizaci databáze byl použit systém MySQL [MySQL]. Tato databáze byla vybrána pro její velmi dobrou přenositelnost mezi systémy MS Windows a Linux.

4.3.2. Struktura databáze

Strukturu databáze můžeme rozdělit do čtyř hlavních částí, které se liší použitím a vztahem k aplikaci.

- ***Informace o uživateli***. V této části jsou uložena data obsahující informace o uživateli. Přistupuje k ní webové rozhranní. Jedná se o data nezbytá pro přihlášení a identifikaci uživatele a uživatelská práva. Tato data jsou uložena v tabulkách *user* a *userTemp*.

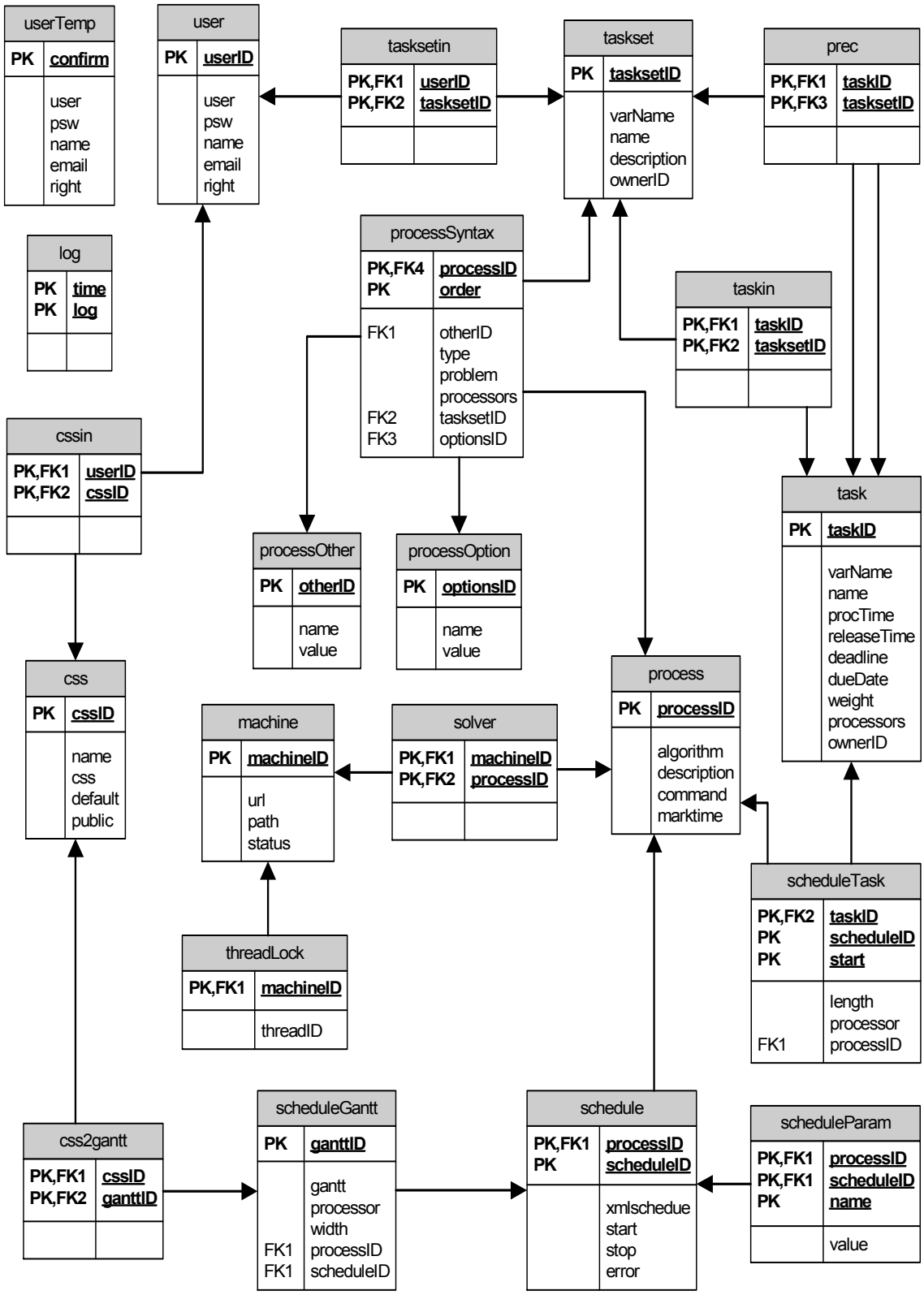
- ***Zadání úlohy***. Uživatelem zadaná data pro rozvržení jsou uložena v této části databáze. Jsou to informace o úlohách, které je potřeba rozvrhnout a informace o zvoleném algoritmu ze scheduling toolboxu. V tabulkách *task, taskset, tasksetin, taskin* a *prec* jsou uloženy všechny parametry úloh a jejich vzájemných vztahů a omezení. V tabulkách *process, processSyntax, processOption* a *processOther* jsou parametry zvoleného algoritmu a jeho syntaxe pro Matlab server. Pomocí této části databáze se předávají data zadaná uživatelem přes rozhranní na Matlab server.

[3] Více o systému zámků v kapitole 4.6.3.

- ***Informace o řešení úlohy***. Pro zabezpečení průběhu řešení úlohy jsou v databázi uchovávána data o právě řešených úlohách a stavu jednotlivých Matlab serverů. Pomocí dat uvedených v těchto tabulkách je zajištěno paralelní zpracování úloh a systém zámků. V tabulce *machine* jsou uloženy adresy a stav Matlab serverů, kterým je pomocí tabulky *solver* přiřazena úloha k řešení. Tabulka *threadLock* slouží jako zámek pro řešení pomocí vícevláken. Dále do této skupiny patří tabulka *log*, kam jsou zapisovány informace o průběhu řešení.

- ***Výsledky úlohy***. Výsledný rozvrh ve formě ganttových diagramů, rozvrhu úloh a hodnot optimalizačních kritérií je uložen v tabulkách *schedule, scheduleTask, scheduleParam* a *scheduleGantt*. S tabulkou scheduleGantt je svázána relací tabulka *css*, která obsahuje kaskádové styly pro vytváření ganttových diagramů pomocí aplikace psGantt. Prostřednictvím této části databáze se přenáší výsledky úloh z Matlab serveru zpět na rozhranní.

Podrobný popis tabulek použité databáze se nachází v příloze A.

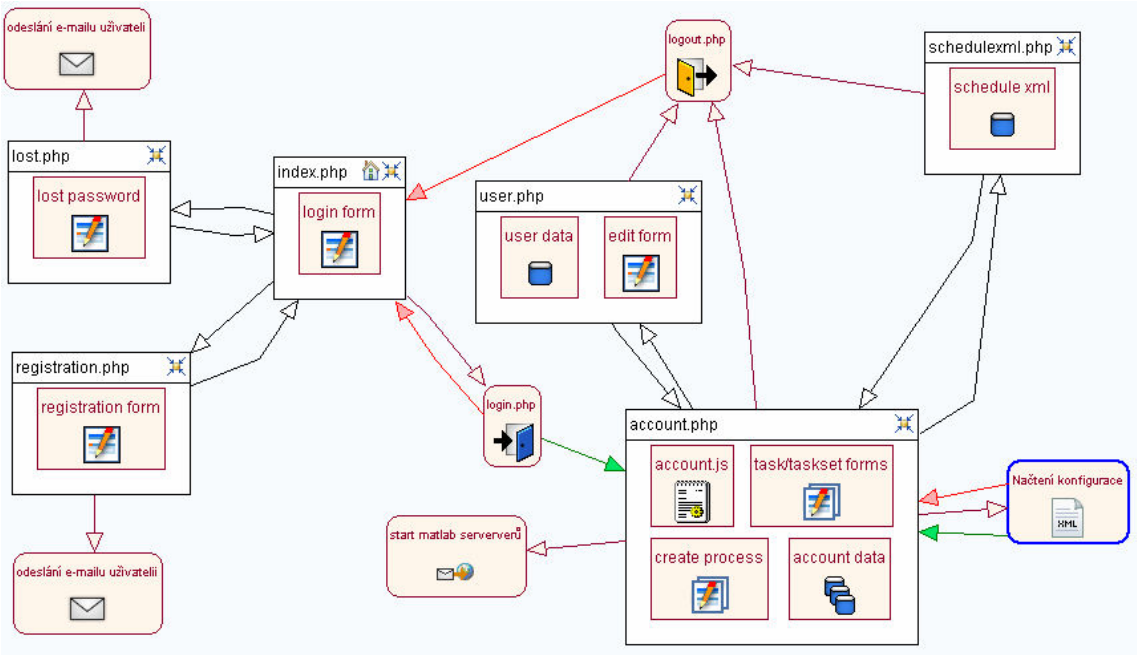


Obr. 4.3: struktura databáze

4.4. Webové rozhranní

Jedinou součástí aplikace se kterou přichází uživatel do přímého kontaktu je webové rozhranní. Webové rozhranní tedy musí uživateli poskytnout veškeré funkce nezbytné pro snadnou a úspěšnu práci s TORSCHE. Pro implementaci je použito technologie PHP a JavaScript. Aby nebylo nutné načítat celou stránku, pokud je potřeba změnit jen její část, je využito konceptu AJAX. Koncepce rozhranní vychází ze systému účtů, které umožňují uživatelům přístup a správu dat. Data potřebná pro rozvrhování a správu rozhranní jsou načítána a ukládána do databáze.

4.4.1. Struktura rozhranní



Obr. 4.4: stuktura rozhranní

Struktura rozhranní je složena ze šesti webových stránek a skriptů pro jejich obsluhu obr.4.4.

- ***Index.php*** – Domovská stránka určená pro přihlášení do aplikace.
- ***Registration.php*** – Stránka s formulářem pro registraci nového uživatele.
- ***Lost.php*** – Pomocná stránka pro uživatele, kteří zapomněli své uživatelské heslo.
- ***Account.php*** – Dynamicky generovaná stránka obsahující nástroje pro práci s uživatelským účtem a pro práci s TORSCHE. Podle hodnot vstupních parametrů metody GET je generován obsah stránky.
- ***User.php*** – Stránka s informacemi o účtu uživatele.
- ***Schedulexml.php*** – Stránka pro zobrazení zozvrhu ve formě XML.

Stránky `index`, `registration` a `lost` jsou určeny pro vytvoření přístupu k práci na uživatelském účtu. Přístup k webovým stránkám `user`, `account` a `schedulexml` je povolen pouze po úspěšném přihlášení. Vzhledem k tomu, že protokol HTML, který je použit pro komunikaci uživatele s webovým serverem, je bezstavý[4] je nutné uchovávat informace o stavu aplikace pomocí proměnných SESSION. Ve stavových proměnných SESSION jsou uchovávány informace o přihlášeném uživateli a jeho právech pro práci s rozhranním.

4.4.2. Komunikace

Web server potřebuje pro správnou funkčnost komunikovat s databází a musí umožňovat spouštění výpočtů na Matlab serveru. Pro komunikaci s databází je využívána knihovna pro PHP *php_mysql*. Spouštění výpočtů na Matlab serveru je zajištěno prostřednictvím CGI. CGI skript na Matlab serveru je možné spustit dvěma způsoby.

- ***Spouštění s odezvou***: Spouštění s odezvou je vytvořeno na skriptu HttpClient [HttPClient], který umožňuje prostřednictvím socketu spoustit skript na Matlab serveru a zobrazit HTML, které vrací. Tento způsob komunikace je určen pro administrátora, aby mohl podrobně sledovat průběh řešení úloh na Matlab serveru.

- ***Spouštění bez odezvy***: Tato varianta je založena na použití socketu, pomocí kterého je iniciován script na Matlab serveru, ale nedochází k čekání na vrácené parametry. Tato varianta je použita při práci běžného uživatele, aby nedocházelo ke zbytečným zdržením při čekání na vrácené HTML.

4.4.2. Konfigurace

Aby funkčnost poskytnutá rozhranním odpovídala práci s toolboxem, je nutné udržovat nabídku algoritmů a parametrů aktuální. K tomuto účelu je použito konfiguračního XML souboru, který obsahuje seznam algoritmů a jejich parametrů včetně vyžadované syntaxe. Ukázka části konfiguračního souboru je na obr. 4.5. Tento konfigurační soubor je v aplikaci zpracován pomocí XML parseru, který z něj vybere potřebné informace. Parser je vytvořen pomocí XML DOM (Document Object Model) funkcí implementovaných v *domxml* rozšíření pro PHP.

[4] HTTP protokol neposkytuje žádné informace o stavu komunikace.

```
  <?xml version="1.0" encoding="UTF-8" ?>
  <?oxygen RNGSchema="algorithm.rnc" type="compact"?>
- <algorithmlist ver="0.1" application="Scheduling Toolbox">
    ...
  - <algorithm id="SPNTL">
      <name>SPNTL</name>
      <command>spntl</command>
      <description>Scheduling with Positive and Negative Time-Lags</description>
      <reference>http://rtime.felk.cvut.cz/scheduling-toolbox</reference>
      <problem type="other" syntax="1">SPNTL</problem>
      <problem type="blazewicz" syntax="1">1|prec(lij)|Cmax</problem>
      <problem type="blazewicz" syntax="1">1|rj,dj|Cmax</problem>
    - <syntax id="1">
        <argument>taskset</argument>
        <argument>problem</argument>
      - <argument type="options">
        - <field>
            <name>spntlMethod</name>
            <description>Specifies a method for SPNTL algorithm</description>
            <default>ILP</default>
            <datatype type="list">BaB|ILP</datatype>
          </field>
        - <field visible="false">
            <name>verbose</name>
            <description>Verbose mod</description>
            <default>2</default>
            <datatype min="0" max="2">int</datatype>
          </field>
        </argument>
      </syntax>
    </algorithm>
    ...
  </algorithmlist>
```

Obr. 4.5: konfigurační XML soubor

4.4.2.1. Popis konfiguračního souboru

Z konfiguračního souboru jsou získána všechna data nezbytná pro použití daného algoritmu. Kořenový element `algorithmlist` obsahuje elementy `algorithm` obsahující popis a syntaxi daného algoritmu. Atribut `id` elementu algorithm popisuje unikátní identifikátor daného algoritmu. Potomci elementu algorithm:

- ***name*** – Tento element obsahuje název algoritmu.
- ***command*** – Tento element obsahuje příkaz, kterým je algoritmus volán v MATLABu, tento údaj je uveden u všech algoritmů.
- ***description*** – Tento element obsahuje popis daného algoritmu.
- ***reference*** – V tomto elementu je uložen odkaz na informace o daném algoritmu.

- ***problem*** **-** Tento element popisuje rozvrhovací problém, který daný algoritmus řeší. Obsahuje parametr `type`, který určuje zda se jedná o problém zapsaný dle standardní notace (Błażewicz), nebo o jiný typ problému. Pokud je možné volat algoritmus pomocí více syntaxí je v atributu `syntax` uvedeno pro jaký typ syntaxe lze tento problém použít. Pokud je možno použít při volání algoritmu více rozvrhovacích problémů typu Błażewicz, obsahuje element problem potomky:
  - **alpha** – parametr α dle standardní notace
  - **betha** – parametr β dle standardní notace
  - **gamma** – parametr γ dle standardní notace

  Atribut `bethamultiple` potom určuje možné kombinace těchto elementů pro vytvoření rozvrhovacího problému.

- ***syntax*** **–** Element `syntax` popisuje syntaxi algoritmu. Atribut `id` určuje identifikátor dané syntaxe. Jeho potomci jsou elementy `argument` popisující vstupní argumenty algoritmu. Definovány jsou argumenty `taskset, problem, processors, options a other`. Element `argument` obsahuje pro agumenty `options` potomky `field`. `Field` popisuje položku daného argumentu, obsahuje potomky:
  - **name –** jméno argumentu
  - **description –** popis argumentu
  - **default –** defaultní hodnota
  - **datatype –** dle obsažených atributů buď obsahuje možné hodnoty argumentu, nebo typ argumentu, kdy rozsah hodnot určují atributy `min` a `max`.

  Attribute visible elementu `field` určuje zda daný argument může uživatel nastavovat. Stejné potomky jako element `field` obsahuje element `argument` pro argument `other`.

4.4.3. Systém účtů

Práce s rozhranní funguje na systému účtů, kde každý uživatel má k dispozici vlastní účet s přístupem chráněným heslem. Na těchto účtech uživatelé vytvářejí úlohy, které chtějí rozvrhnout pomocí TORSCHE.

4.4.3.1. Založení účtu

Založení účtu nového uživatele se skládá ze dvou kroků. Nejprve se uživatel zaregistruje a poté účet aktivuje prostřednictvím odkazu, který obdrží na svou e-mailovou adresu. Při registraci je od uživatele vyžadováno vytvoření unikátního uživatelského jména a hesla, která bude používat pro přístup k osobnímu účtu. Registrace nového uživatele se provádí prostřednictvím formuláře umístěného na webové stránce `registration.php` obr. 4.6.

Registration form

Login name:

Name:

Password:

Retype password:

E-mail:

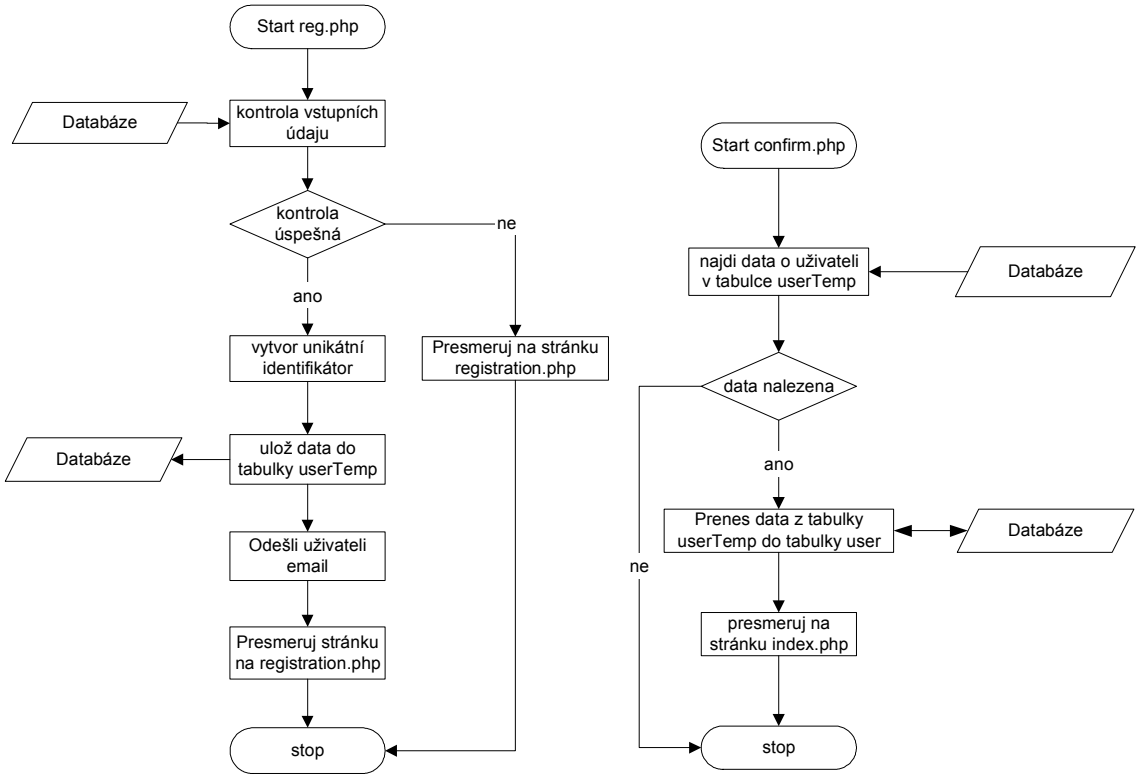
Submit Reset

Obr. 4.6: registrační formulář

Vývojové diagramy průběhu registrace jsou na obr. 4.7. Uživatelem zadané údaje jsou odeslány metodou POST do skriptu `reg.php` pro zpracování. Ve skriptu jsou nejprve zkontrolovány vstupní údaje. Kontroluje se unikátnost uživatelského jména, tvar emailové adresy a zda si odpovídají obě zadaná hesla. Pokud není test úspěšný, je uživatel přesměrován zpět na stránku `registration.php` spolu se zachycenou chybou. V opačném případě je pro uživatele vytvořen unikátní 32 znakový identifikátor a data jsou uložena do tabulky `userTemp`. Uživatelské heslo je z bezpečnostních důvodů před uložením do databáze transformováno metodou salted hash [Interval] za použití hešovací funkce MD5. Na e-mailovou adresu je uživateli poslán odkaz na webovou stránku `confirm.php` s parametrem `id`, který odpovídá identifikátoru klienta:

```
webserveraddress/confirm.php?id=c22d9317e23527638b4b7f4dc0ddd1dc
```

Aktivace účtu je vykonávána na webové stránce `confirm.php`, kam se uživatel dostane po kliknutí na obdržený odkaz. Skript podle proměnné `id` najde v tabulce `userTemp` data, která uživatel zadal a přenese je do tabulky `user`. Tímto je účet aktivovaný a uživatel je přesměrován na stránku `index.php`, kde se může přihlásit.



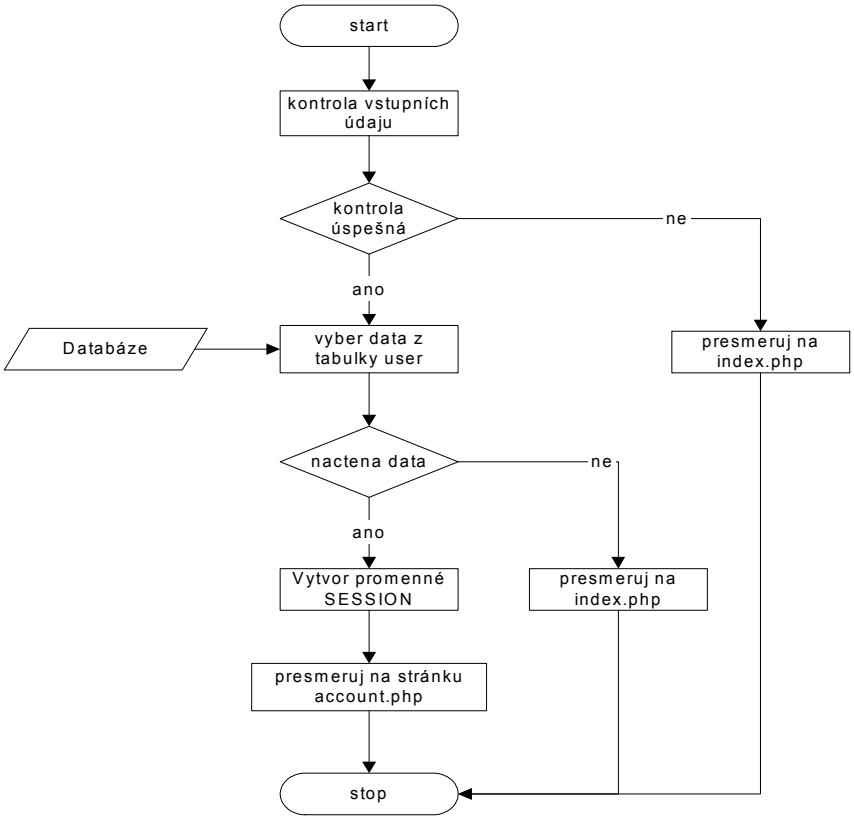
Obr. 4.7: registrace uživatele vývojové diagramy

4.4.3.2. Přihlášení a odhlášení

Pokud chce uživatel rozhranní používat, musí se přihlásit pomocí platného uživatelského jména a hesla, nebo využít přihlášení bez registrace jako host. Pro přihlášení slouží na stránce `index.php` umístěný formulář obr. 4.8, do kterého uživatel vyplní své přihlašovací údaje.

Obr. 4.8: přihlášení uživatele

Vyplněné údaje z formuláře jsou odeslány prostřednictvým metody POST a zpracovány pomocí skriptu `login.php` obr.4.9. Po zkontrolování vstupních údajů se skript pokusí pomocí SQL příkazu načíst z databáze řádek obsahující zadané uživatelské jméno a heslo. Heslo je před použitím hešováno pomocí metody salted hash se stejnými parametry, které byly použity při registraci. Pokud SQL příkaz najde odpovídající řádek jsou vytvořeny proměnné SESSION s informacemi o uživateli, které slouží pro ověření, že je uživatel přihlášen. Přihlášený uživatel je přesměrován na stráku `account.php`. Odhlášení uživatele zajišťuje skript `logout.php`, který zruší proměnné SESSION a přesměruje komunikaci zpět na webovou stránku `index.php`.



Obr. 4.9: přihlášení uživatele vývojový diagram

4.4.3.3. Uživatelská práva

Operace, které může uživatel na rozhranní vykonávat jsou omezeny jeho uživatelskými právy. Rozhranní podporuje tři druhy uživatelských práv.

- **admin** - administrátorská práva. Práce s rozhranním není nijak omezena, uživatel má právo zasahovat i do účtů ostatních uživatelů a má k dispozici administrátorské funkce.
- **user** – standardní uživatelská práva. Práce s rozhranním je omezena jen pro obsluhu jednoho účtu, ke kterému je uživatel přihlášen.
- **guest** – host. Host je uživatel, který může vykonávat stejné operace jako uživatel user, všechna jeho data jsou po ukončení práce s rozhranním vymazána.

Každý uživatel, který se do aplikace zaregistruje má standardně přiděleno uživatelské právo typu *user*, toto právo mu může zmenit pouze uživatel s právem *admin*. Počet uživatelů s uživatelskými právy *admin, user* a *guest* není nijak omezen.

4.4.3.4. Zapomenuté heslo

V případě zapomenutí hesla pro přístup k osobnímu účtu umožňuje rozhranní jeho změnu. Heslo není možné uživateli připomenout nebo poslat na e-mail z toho důvodu, že uživatelská hesla se v databázi z bezpečnostních důvodů nikde neukládají, ale ukládají se pouze jejich

otisky. O změnu hesla v případě jeho zapomenutí je možné zažádat na webové stránce `lost.php`. Na této stránce je k dispozici formulář do kterého uživatel vyplní uživatelské jméno nebo emailovou adresu, kterou použil pro registraci obr.4.10.

Lost password
User name or email: Send me password
(Info for your password recovery will be sent to your email address.)

Obr. 4.10: formulář pro zapomenuté heslo

Aplikace ověří data a vygeneruje unikátní 32 znakový identifikátora pro tohoto uživatele. Data o uživateli z tabulky `user` se uloží spolu s identifikátorem do tabulky `userTemp` a uživateli se pošle e-mail s webovou adresou stránky `confirm.php`, který obsahuje parametr s identifikátorem uživatele:

```
webserveraddress/confirm.php?code=c22d9317e23527638b4b7f4dc0ddd1dc
```

Tímto je zajištěno, že i nadále funguje přihlášení pomocí starého uživatelského hesla a změnu zapomenutého hesla může uskutečnit pouze osoba, která má přístup k e-mailu. Stránka `confirm.php` slouží pro přístup ke změně hesla, uživatel je identifikován pomocí identifikačního kódu, který získal při žádosti o změnu. Identifikátor je porovnán s identifikátory uloženými v tabulce `userTemp`. Pokud je identifikace uživatele úspěšná je aplikací přihlášen a přesměrován na stránku `user.php`, kde si může heslo změnit.

4.4.4. Práce s rozhranním

Po přihlášení na účet je uživateli zobrazen přehled informací o jeho účtu obr. 4.11.

Click here to create new set of tasks.

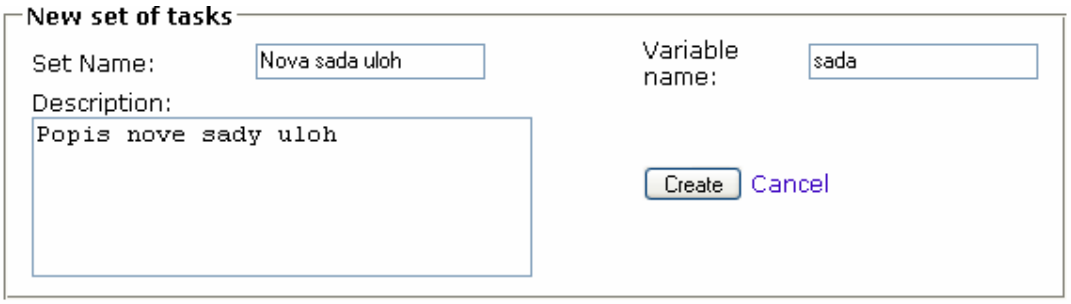
Set name	tasks	processes:	solved	not solved	error
Nova sada uloh	4	3	3	0	0
setoftasks1	3	0	0	0	0
Vyroba nabytku	9	0	0	0	0
Sum	**16**	**3**	**3**	**0**	**0**

Obr. 4.11: přehled účtu

V tomto přehledu je odkaz na formulář pro založení nové sady úloh a tabulka s přehledem existujících sad úloh. U každé sady je uveden počet úloh, které obsahuje, počet zadaných úloh, počet vyřešených úloh, počet nevyřešených úloh a počet úloh, jejichž výpočet vrátil chybu. Tato tabulka zároveň slouží jako rozcestí pro další práci. Kliknutím na příslušné políčko tabulky je uživateli zobrazena příslušná stránka.

4.4.4.1. Sady úloh

Pro vytváření sady úloh slouží formulář obr. 4.12.

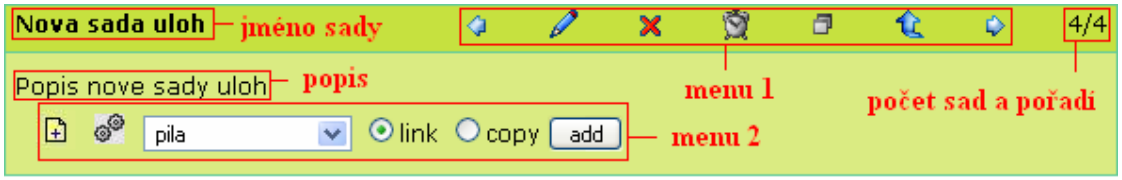


Obr. 4.12: formulář pro vytvoření sady úloh

Vstupní parametry pro vytvoření nové sady úloh jsou:

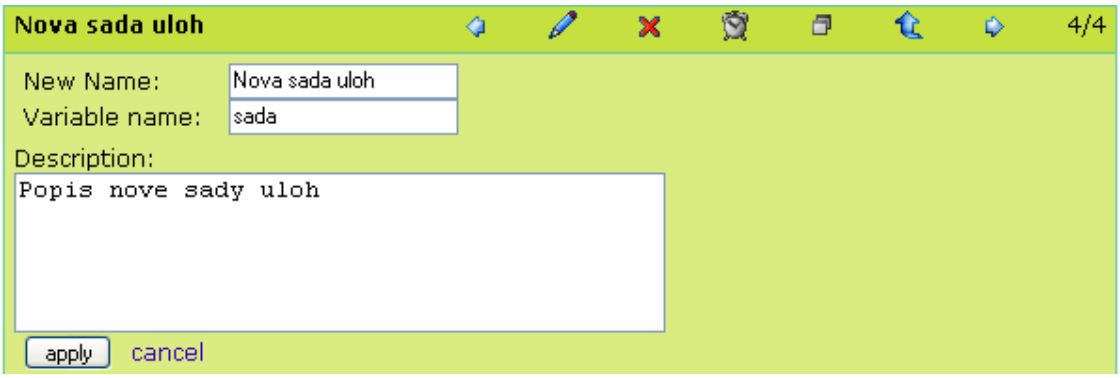
- **set name** – jméno sady úloh
- **variable name** - jméno použité pro název proměnné v které bude uložen objekt typu `taskset` při řešení v MATLABu
- **description** – popis sady úloh pro lepší orientaci uživatele

Všechny zadávané parametry je nutné vyplnit. Takto vytvořená sada úloh je v databázi uložena v tabulce `taskset`. Na obrázku obr. 4.13 je zobrazena nově vytvořená sada úloh. Sada úloh je zobrazena jako panel v jehož hlavičce se nachází jméno vytvořené sady, menu 1 pro operace se sadou úloh a pořadí sady. V těle panelu je popis úlohy a menu 2 pro vytváření a přidávání úloh.



Obr. 4.13: sada úloh

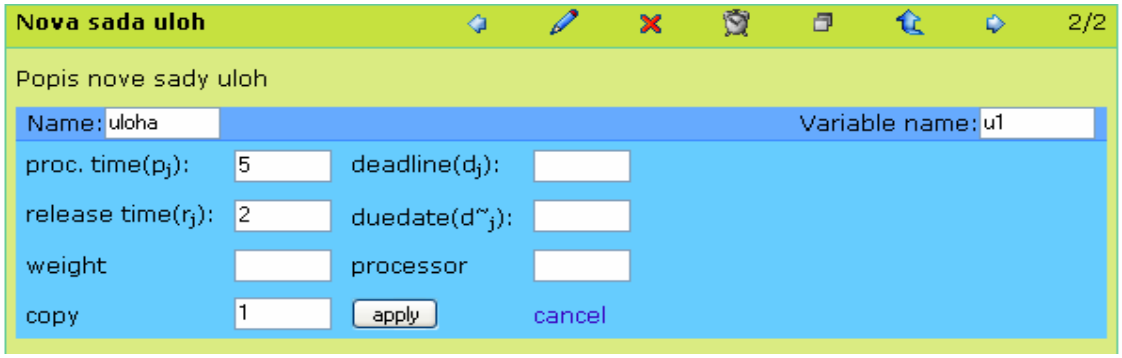
Pomocí menu 1 je možné listovat mezi jednotlivými sadami úloh, editovat parametry dané sady (jméno, jméno proměnné, popis), smazat sadu, zvolit algoritmus a problém pro rozvrhování, nastavit precedenční omezení a návrat do přehledu informací o účtu.



Obr. 4.14: editace sady úloh

4.4.4.2. Úlohy

Do sady úloh může být přidána zcela nová úloha, kopie úlohy dané sady nebo kopie již existující úlohy z jiné sady úloh, kterou uživatel vlastní. Nová úloha se vytváří pomocí formuláře obr. 4.15.

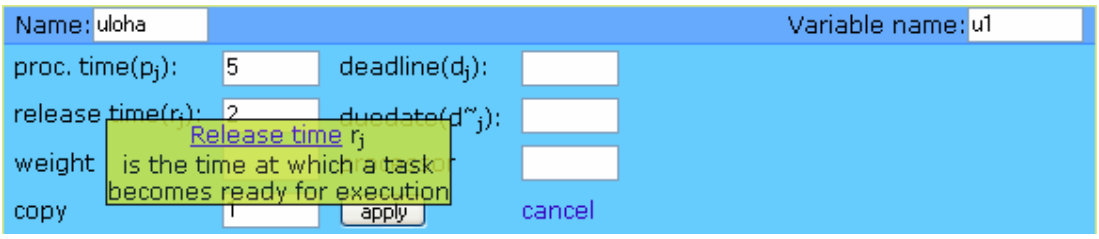


Obr. 4.15: formulář pro vytvoření úlohy

Vstupní parametry pro vytvoření nové úlohy jsou:

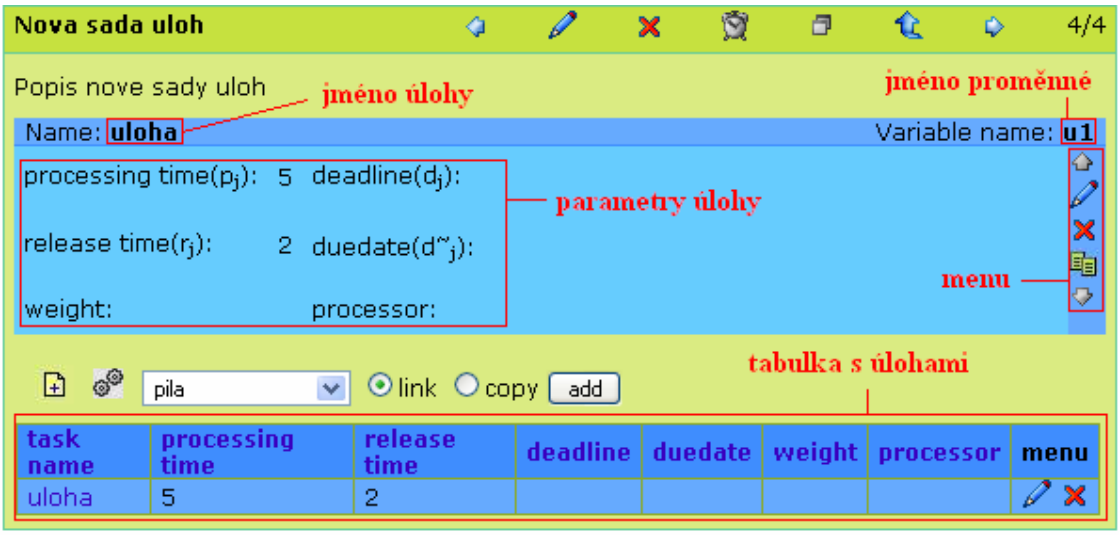
- **Name** – název úlohy
- **Variable name** – název proměnné reprezentující úlohu při řešení v MATLABu
- **proc. time** – doba vykonávání úlohy
- **release time** – okamžik disponibility
- **deadline** – poslední okamžik dokončení
- **duedate** – okamžik požadovaného dokončení
- **weight** – priorita úlohy
- **processor** – číslo procesoru na kterém se musí úloha vykonávat
- **copy** – kolikrát se má úloha vytvořit

Povinné údaje, které musí být vyplněny, jsou `Name`, `Variable name`, `proc. time` a `copy`. Pro snadnější orientaci v jednotlivých parametrech úlohy, zobrazuje se nápověda, při ukázání kurzorem myši na příslušný parametr. Toto ovšem funguje pouze se zapnutým JavaScriptem. Na obrázku obr.4.16 je ukázka nápovědy pomocí JavaScriptu.



Obr. 4.16: nápověda parametrů úlohy

Vytvořené úlohy jsou na rozhranní zobrazeny dle obr. 4.17. Zobrazí se detail úlohy obsahující jméno úlohy, jméno proměnné, parametry úlohy a menu pro operace s úlohou. Dále se zobrazí tabulka obsahující všechny úlohy dané sady. Pomocí menu úlohy je možné listovat mezi jednotlivými úlohami, editovat parametry, smazat úlohu nebo vytvořit kopii úlohy.



Obr. 4.17: sada úloh s úlohou

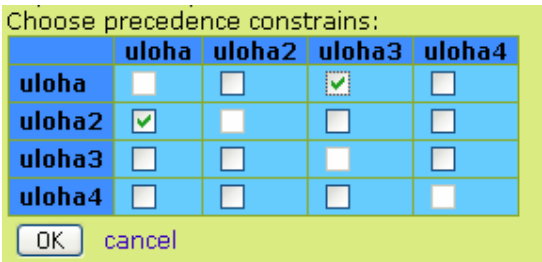
Úlohy mohou být do sady úloh také přidány z jiných sad pomocí formuláře obr. 4.18. Pomocí select boxu si uživatel vybere úlohu, kterou chce přidat a volbou link nebo copy určí zda jí chce pouze přilinkovat nebo přímo zkopírovat.

Obr. 4.18: formulář pro přidání úlohy

Poslední možností jak přidávat úlohy do sady úloh je vytvořit kopii úlohy umístěné v této sadě. K tomu slouží pouhé kliknutí na ikonu pro vytvoření kopie v menu úlohy.

4.4.4.3. Zadání precedenčních omezení

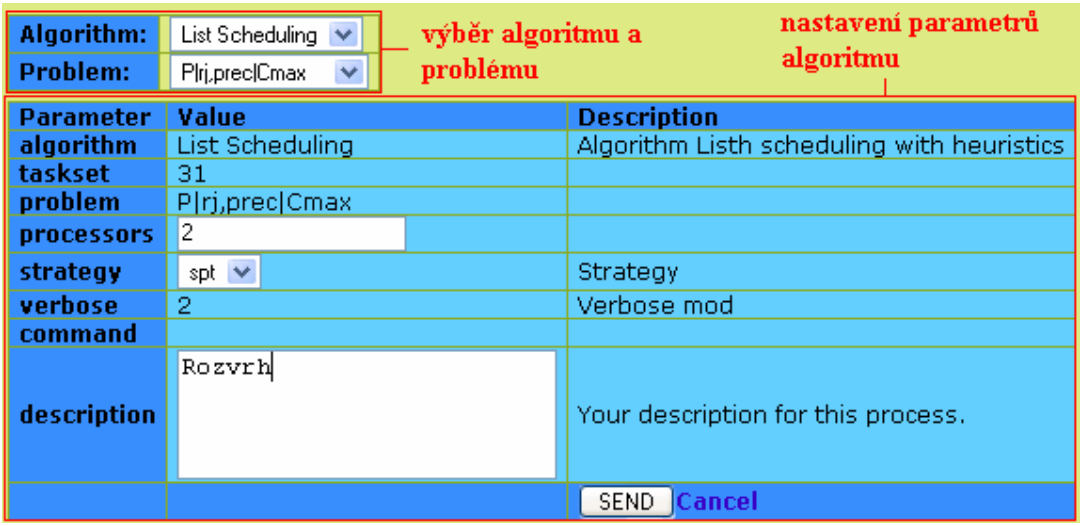
Vytváření precedenčních omezení v rámci sady úloh je prostřednictvím formuláře obr. 4.19. Závislosti mezi úlohami jsou zadávány do tabulky, která reprezentuje matici precenečních omezení, pro kterou platí stejná pravidla jako v TORSCHE kap.3.2.1.



Obr. 4.19: formulář pro vytváření precedenčních omezení

4.4.4.4. Výběr algoritmu

Sadu úloh s alespoň dvěma úlohama můžeme nechat rozvrhnout pomocí zvoleného algoritmu obsaženého v TORSCHE. Pro rozvrhování sady úloh slouží dynamicky generovaný formulář obr. 4.20, který dle zvoleného algoritmu a problému vytvoří formulář s vyžadovanými parametry. Formulář je generován na základě údajů v konfiguračním XML souboru.

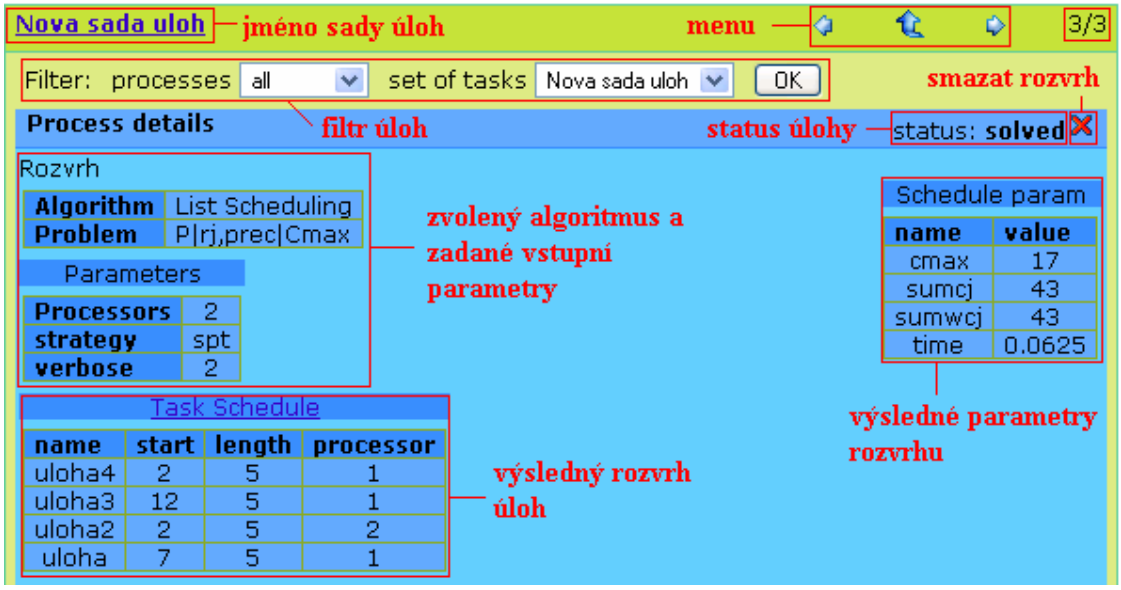


Obr. 4.20: rozvrhování sady úloh

Odesláním tohoto formuláře je spuštěn skript, který data zkontroluje a uloží do databáze. Po uložení dat do tabáze jsou spuštěny Matlab servery pro vyřešení zadaných úloh.

4.4.4.5. Rozvrh

Pro zobrazení naplánovaných nebo vyřešených úloh je vygenerována nová stránka obsahující panel sady úloh. V hlavičce panelu se nachází jméno sady úloh která byla rozvržena, menu pro listování rozvrhama pro tuto sadu a ukazatel pořadí zobrazeného rozvrhu. V těle panelu se nachází panel s detaily rozvrhu a tabulka se seznamem všech zadaných úloh pro tuto sadu. Panel s detaily rozvrhu obsahuje v hlavičce položku status, která určuje v jakém stavu se úloha nachází (zda je úloha naplánovaná pro vyřešení, řešená, úspěšně vyřešená nebo neúspěšně vyřešená). V těle panelu jsou zobrazeny parametry zadání rozvrhu, parametry řešení a ganttův diagram. Na obrázku obr.4.21 je zobrazena horní část stránky s rozvrhy.

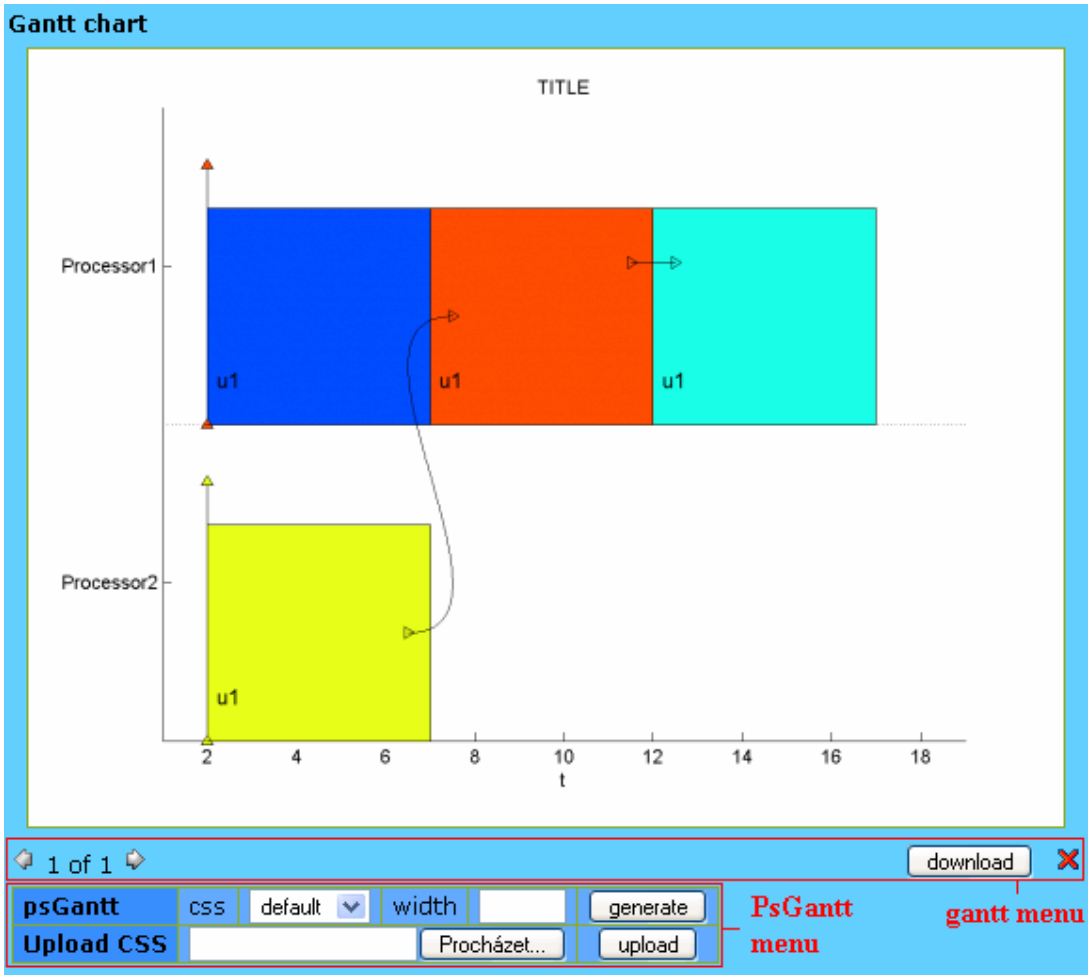


Obr. 4.21: rozvrh

4.4.4.6. PsGantt

V panelu s detailním popisem rozvrhu úlohy obr. 4.22. se nachází spolu s ganttovým diagramem, který byl vytvořen při řešení v MATLABu, i menu umožňující generovat ganttovy diagramy pomocí nástroje PsGantt. PsGantt menu obsahuje dva formuláře. Jeden formulář slouží pro nahrávání kaskádových stylů, které popisují zobrazení diagramu. Nahrávané kaskádové styly musí mít koncovku css, jinak je uložení do databáze zamítnuto. Pomocí druhého formuláře můžeme gantův diagram vygenerovat. Vstupními parametry jsou použitý soubor kaskádových stylů, který vybereme z nahraných souborů pomocí select boxu, a šířka výsledného ganttova diagramu. Odesláním těchto parametrů je zavolána webová služba PsGantt a výsledný diagram je uložen do databáze. Menu umístěné pod ganttovým diagramem umožňuje listovat ganttovými diagramy náležící úloze a ukládat je na disk. Ganttovy diagramy je možné stahovat ve formátech png, diagramy generované webovou službou PsGantt můžeme stahovat ve formátech, které tato služba nabízí. V současné době jsou k dispozici formáty PDF[5], PNG[6] a EPS[7].

[5] Portable Document Format
[6] Portable Network Graphic
[7] Encapsulated PostScript



Obr. 4.22: ganttovy diagramy

4.4.4.8 Administrátor

Administrátor má k dispozici speciální menu obr.4.23, které mu umožňuje zasahovat do účtů ostatních uživatelů a kontrolovat funkčnost aplikace.

Obr. 4.23: menu administrátora

Menu obshuje tři položky:

- **log** – Pomocí této položky si administrátor zobrazí průběhy řešení úloh obr.4.24, které jsou uloženy v tabulce **log**.
- **switch user** – Tato položka umožňuje administrátorovi přistupovat k rozhranní jako jiný uživatel a zasahovat tak do jeho účtu. Tato možnost je zajištěna pomocí speciální proměnné administrátora `$_SESSION['altuser']`, ve které je uložen identifikátor jiného uživatele. Po ověření identity administrátora je tato proměnná použita jako identifikátor přihlášeného uživatele, který umožňuje obsluhu daného účtu.

- **delete user** – Pomocí této položky může administrátor mazat účty jednotlivých uživatelů.

log	switch user	delete user
hide	test SWITCH	delete user martij

2007-05-21 10:14:00	Machine 2 assigned process 1, thread locked
2007-05-21 10:14:01	Machine 2 process 1 locked
2007-05-21 10:14:04	Machine 2 no process assigned
2007-05-21 10:14:04	Machine 2 process 1 solution saved, locks removed
2007-05-21 10:24:27	Machine 2 assigned process 2, thread locked
2007-05-21 10:24:28	Machine 2 process 2 locked
2007-05-21 10:24:30	Machine 2 no process assigned
2007-05-21 10:24:30	Machine 2 process 2 solution saved, locks removed
2007-05-21 10:46:25	Machine 2 assigned process 3, thread locked

Obr. 4.24: zobrazení informací o průběhu řešení úlohy

4.4.4.7 Zobrazení chyb a varování

Pokud je na rozhranní provedena neplatná operace je uživateli zobrazeno chybové hlášení. Na obrázku obr. 4.25 je příklad chybového hlášení, které se zobrazí pokud neuvedeme jméno vytvářené úlohy.

Task name must be filled!

Obr. 4.25: chybové hlášení

Pokud chceme vymazat nějaká vytvořená data, zobrazí se varování vyžadující potvrzení operace obr. 4.26.

Do you realy want to delete Nova sada uloh? All processes and tasks related to this set will be lost. ok cancel

Obr. 4.26: varování

Pokud má uživatel zaplý JavaScript jsou tato hlášení zobrazována pomocí vyskakujících oken.

4.5. Matlab server

Matlab server je výpočetním serverem pro řešení rozvrhovacích problémů. Hlavní součástí Matlab serveru je výpočetní prostředí MATLAB. Pro řešení úloh je použit TORSCHE Scheduling Toolbox pro Matlab. Úkolem Matlab serveru je vyřešit úlohy, které uživatel zadává prostřednictvím webového rozhranní. Jednotlivé úlohy jsou brány z databáze, kam byly uloženy pomocí rozhranní[8]. Komunikace s databází probíhá prostřednictvím SOAP serveru[9] s využitím webových služeb. Jedná se o komunikaci klient-server, kde Matlab server vystupuje v roli klienta, který volá služby SOAP serveru. Důležitým úkolem Matlab serveru

[8] Viz kapitola 4.4.4.4.
[9] Více o SOAP serveru v kapitole 4.6.

je, aby byly úlohy vykonávány co nejrychleji. Z toho důvodu je na serveru řešena v jeden okamžik jen jedna úloha. Tím je umožněno maximální využití výpočetní kapacity serveru pro výpočet. Aby bylo řešení úloh plynulé a rychlé, je pro výpočet více úloh použito více Matlab serverů, které řeší různé úlohy ve stejný okamžik. Pro jednodušší diagnostikování případných chyb při řešení úloh, jsou informace o prováděných operacích na serveru ukládány do log souboru m`atlab_server.log`.

4.5.1. Řešení úloh

Řešení úloh je zajištěno M-souborem SOAPeval, který je inicializován z webového rozhranní prostřednictvím CGI skriptu. M-soubor je na serveru vykonávan cyklicky dokud jsou mu přidělovány nevyřešené úlohy z databáze, nebo dokud není jeho chování serverem vyhodnoceno jako chybné obr.4.27. Běh jednoho cyklu se skládá z pěti hlavních kroků.

1. **inicializace** – Během inicializace získá skript identifikátor úlohy připravené k vyřešení a seznam skriptů potřebných pro její vyřešení. Pomocí tohoto identifikátoru se Matlab server dále identifikuje při komunikaci se SOAP serverem. Pokud skript nezíská identifkátor úlohy, je ukončen.

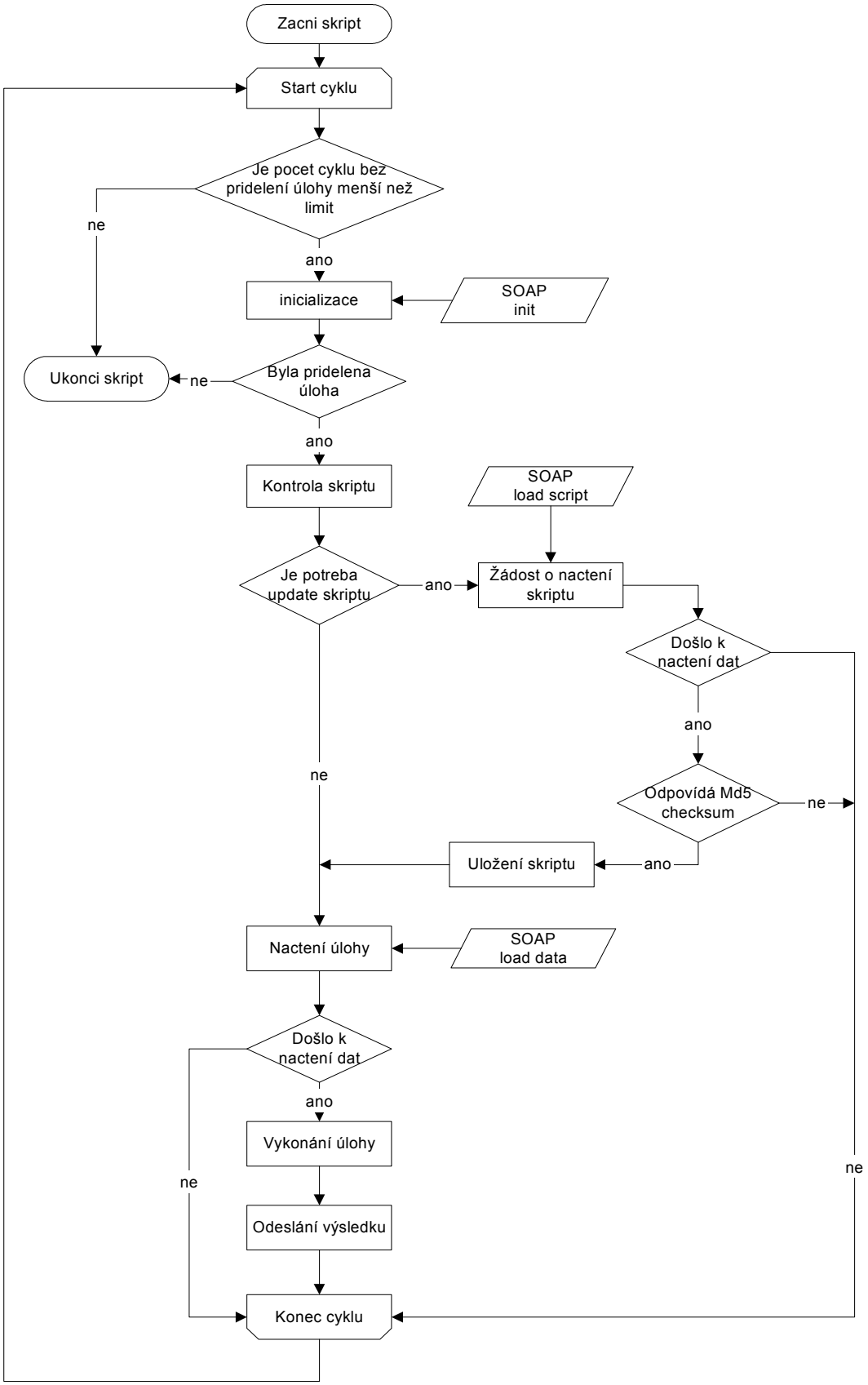
2. **kontrola skriptů –** Provede se kontrolu skriptů potřebných pro řešení přidělené úlohy. Kontroluje se zda daný skript je k dispozici a také, kdy byl jeho obsah změněn. Pokud se nějaký skript nenachází na serveru, nebo je starší verze než ta, kterou nabízí SOAP server, je zažádáno o načtení skriptů ze SOAP serveru.

3. **načtení úlohy –** Pokud je vše připraveno pro rozvržení dané úlohy skript zažádá o zaslání kódu pro její vyřešení. Spolu s kódem pro výpočet rozvrhu získá Matlab server také seznam návratových proměnných jejichž hodnoty mají být vráceny SOAP serveru.

4. **vyřešení úlohy –** Data načtená v předchozím kroku se skript pokusí vyřešit jako sadu funkcí.Řešení je hlídáno pomocí try-catch funkcí pro odchycení chyb.

5. **odeslání výsledků –** Posledním krokeme cyklu je odeslání výsledků. Pokud rozvrhování proběhlo bez problémů jsou odeslány proměnné, které SOAP server vyžadoval. V případě, že se při rozvrhování vyskytly problémy je odesláno chybové hlášení.

Jak je z popisu skriptu patrné, skript je vytvořen tak, aby nebyl závislý na úloze, kterou řeší. Matlab server je tedy možné použít i pro jiné aplikace pouze změnou SOAP serveru, nebo jeho parametrů.



Obr. 4.27: Matlab server vývojový diagram

4.5.2. Komunikace

Komunikace mezi Matlab serverem a SOAP serverem probíhá ve formátu XML prostřednictvím protokolu SOAP. SOAP zprávy umožňují přenos dat ve formátech boolean, int, double, string, struct a anytype. Pro přenos dat prostřednictvím SOAP zpráv jsem zvolil datový typ string do kterého se dají převést ostatní datové typy serializací[10].

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:n="urn:ServerServices"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soap:Body
soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<n:init><sdata>a:1:{s:9:"interpret";s:6:"matlab";}</sdata></n:init>
</soap:Body></soap:Envelope>
```

kód 4.1: SOAP zpráva

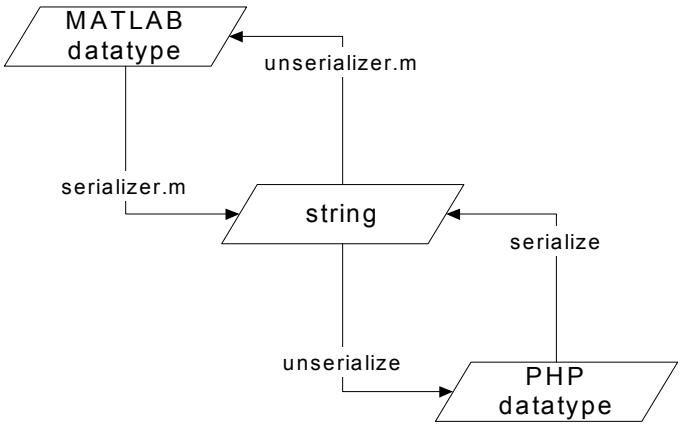
Serializace a deserilizace jsou v PHP implementovány funkcemi `serialize` a `unserialize`. V MATLABu je potřeba tyto funkce vytvořit. Protože se v PHP a MATLABu používají rozdílné datové typy, je nutné zajistit vhodný převod mezi nimi. Tedy datový typ v PHP, který se převede serializací na řetězec znaků musíme být schopni převést deserializací na datový typ v MATLABu a naopak. Proto je nutné najít takové odpovídající datové typy, které si jsou svou logickou strukturou nejbližší. V tabulce 4.1. jsou uvedeny datové typy prostředí MATLAB a jim logickou strukturou nejbližší datové typy v PHP.

MATLAB	PHP
char	string
integer	integer
double scalar	double
logical scalar	logical
cell	Array[index]
struct	Array[key]

Tabulka 4.1: tabulka převodů datových typů PHP-MATLAB

Serializační a deserializační funkce pro MATLAB jsem implementoval funkcemi `serializer` a `unserializer`. Diagram znázorňující převod datových typů mezi PHP a MATLABem je na obr. 4.28.

[10] Serializace je proces, kdy jsou obecná data převedena do proudu dat tak, aby mohla být převedena zpět do původní podoby pomocí opačného procesu. Opačný proces se jmenuje deserializace.



Obr. 4.28: převod datových typů PHP-MATLAB

4.5.3. Inicializace Matlab serveru

Matlab server je nezávisle pracující část aplikace, proto je potřeba zajistit jeho spuštění v době, kdy je to potřeba. Matlab server je inicializován z webového rozhranní prostřednictvím HTML. Původním záměrem bylo využít pro inicializaci skriptů na MATLAB serveru MATLAB Web Server. Vzhledem k tomu, že MATLAB Web Server není od verze R2006b dále podporován a není součástí distribuce, byla pro komunikaci navržena také aplikace Remote run pro vzdálenou inicializaci.

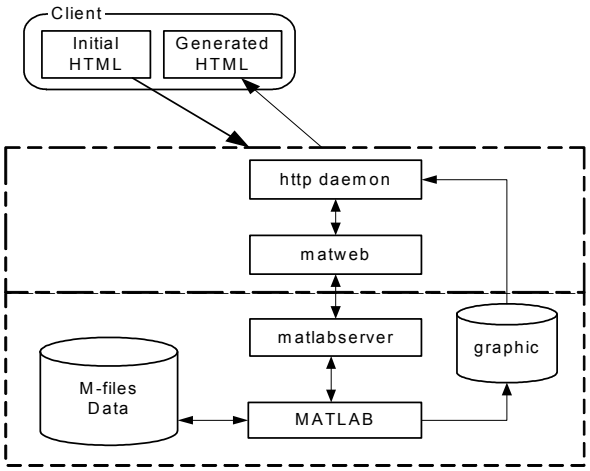
4.5.3.1. MATLAB Web Server

MATLAB Web Server (dále jen MWS) (obr.4.29) je aplikace pro zprostředkování přístupu k MATLAB aplikacím prostřednictvím World Wide Web.

MWS je složen ze sady programů, které umožňují vytvoření aplikací přístupných přes WWW.

- **matlabserver:** Vícevláknový TCP/IP server, který řídí komunikaci mezi Webovou aplikací a MATLABem a spouští M-soubor, který uživatel specifikuje v proměnné mlmfile.
- **matweb:** TCP/IP klient používající CGI pro zpracování HTML dokumentů, které posílá programu **matlabserver**.

[Mathworks]



Obr. 4.29: MATLAB Web Server komunikace

MWS tedy funguje na principu, kdy na bežícím MATLABu jsou prostřednictvím `matlabserveru` spouštěny M-soubory. MWS vyžaduje vstupní a výstupní dokumenty ve tvaru HTML nebo XML, které jsou zpracovány skriptem `matweb`. Vstupní dokument obsahuje proměnnou `mlmfile`, která obsahuje název M-souboru, který se má vykonat a vstupní proměnné. Ve výstupním dokumentu definovanujeme místa, kam se mají zobrazit hodnoty výstupních proměnných.

Pro inicializaci prostřednictvím MWS jsem vytvořil M-soubor `torsche`. Vykonání skriptu `torsche.m` na MWS tedy zavoláme pomocí HTML metody GET:

```
serveraddress/cgi-bin/matweb.exe?mlmfile=torsche
```

M-soubor, který je zavolán má předem definovanou strukturu, pro komunikaci s MWS. Vstupem je struktura `instruct`, vytvořená MWS a výstupem je řetězec `retstr`, který obsahuje výstupní HTML. K jednotlivým vstupním proměnným se přistupuje prostřednictvím vstupní struktury `instruct`.

```
variable=instruct.variablename
```

Výstupní proměnné, které se mají zobrazit ve výstupním HTML se ukládají do proměnné `outstruct`.

```
outstruct.variablename=varible
```

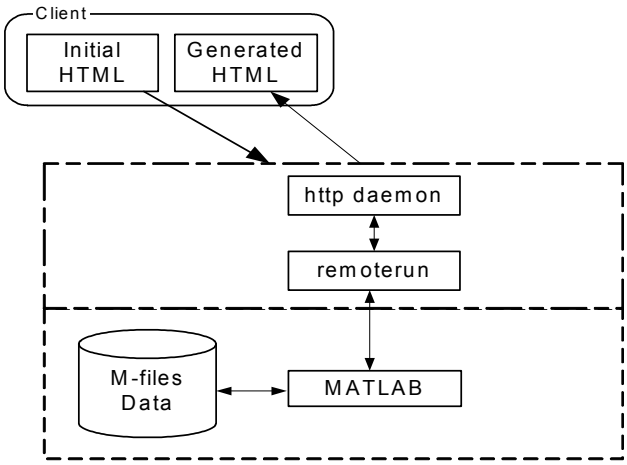
Pomocí funkce `htmlrep` jsou do výstupního HTML `outfile` umístěny výstupní proměnné ze struktury `outstruct`.

```
retstr=htmlrep(outstruct,outHTML)
```

Uvnitř M-souboru `torsche` je zavolán skript `torscheSOAPeval`, který vykonává řešení úloh.

4.5.3.2. Remote run

Remote run (obr.4.30) je aplikace, která zprostředkovává spuštění programu MATLAB a skriptu na serveru prostřednictvím WWW. Aplikace je umístěna na webovém serveru Apache s podporou CGI skriptů.



Obr. 4.30: Inicializace prostřednictvím Remote run

Narozdíl od MATLAB Web Serveru, který spouští M-soubory na běžícím MATLABu, tato aplikace je shellový[11] skript, který spouští celý program MATLAB a M-soubor pomocí příkazové řádky. Shellový skript `remoterun` je spouštěn prostřednictvím CGI. Příkaz pro spuštění skriptu z WWW:

```
serveraddress/cgi-bin/remoterun.sh
```

Skript nemá žádné vstupní parametry, výstupem je HTML dokument obsahující informace o průběhu rozvrhování. Uvnitř skriptu je spuštěn MATLAB a v něm M-soubor `remoterun`. Příkaz pro vzdálené spuštění:

```
/usr/local/matlab73/matlab -nodisplay -nosplash -r remoterun
```

Tímto příkazem je z příkazové řádky spuštěn program MATLAB bez grafického rozhranní. Aby nedocházelo k tomu, že MATLAB zůstane po vykonání skriptu aktivní, je před ukončením vykonávání M-souboru `remoterun` MATLAB ukončen příkazem `quit`.

4.6. SOAP server

SOAP server je vytvořen pro zprostředkování komunikace mezi výpočetním serverem a databází. Vrámci této komunikace server zajišťuje předzpracování dat, aktualizaci algoritmů, ošetření kolizí a chyb. SOAP server nabízí služby pro komunikaci výpočetního prostředku, v našem případě Matlab serveru, s databází. Jedná se o služby implementovné v jazyku PHP, s využitím balíčku SOAP [PEAR]. Komunikace není závislá na typu výpočetního prostředku (klienta). Výpočetní prostředek při komunikaci se SOAP serverem definuje svůj typ (matlab,

[11] Shellový skript komunikuje s operačním systémem prostřednictvím příkazové řádky.

c, c++, aj.) pomocí proměnné `interpret`. Podle hodnoty této proměnné server vykonává skripty určené pro daný typ výpočetního prostředku. V našem případě byly na serveru implementovány skripty pro komunikaci s programem typu MATLAB, který se identifikuje identifikátorem `interpret` s hodnotou matlab. Rozšíření o další typy výpočetních prostředků je velmi jednoduché a vyžaduje pouze přidání skriptů pro jejich obsluhu.

4.6.1. Nabízené webové služby

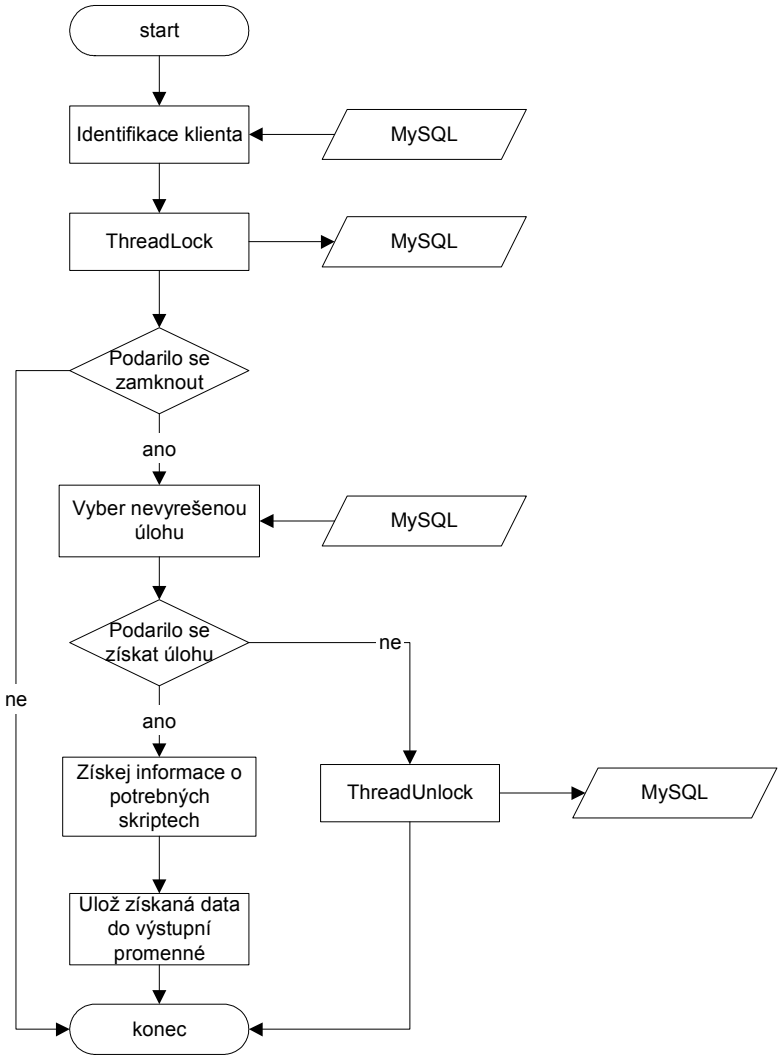
SOAP server nabízí pro komunikaci výpočetního prostředku s databází tuto sadu webových služeb, které jsou implementovány pro výpočetní prostředek MATLAB:

- **init** – inicializace komunikace, přidělení úlohy
- **loadscript** – načtení skriptů potřebných pro řešení úlohy
- **loaddata** – načtení úlohy z databáze a generování kódu pro řešení úlohy
- **senddata** – uložení výsledků úlohy do databáze
- **serializetest** – test komunikace

Init

Init je služba, která zajišťuje inicializaci komunikace mezi Matlab serverem a databází. SOAP server ověří identitu vzdáleného Matlab serveru, který o službu žádá a vyhledá v databázi nevyřešenou úlohu, která je první ve frontě pro vyřešení. Vstupem funkce *init* je identifikátor, který určuje typ výpočetního prostředku. Výstupem je identifikátor nalezené úlohy a seznam skriptů potřebných pro jeho vyřešení, nebo hodnota false pokud nebyla nalezena žádná úloha připravená pro rozvržení. Průběh vykonávání funkce *init* je na obrázku obr.4.31. Najprve dojde k ověření klienta. Identifikace klienta je ověřena porovnáním proměnné obsahující vzdálenou adresu klienta $_SERVER['REMOTE_ADDR'] s adresami Matlab serverů uloženými v databázi. Pokud je identifikace úspěšná je uzamčena možnost další inicializace tohoto klienta zámkem `ThreadLock`[12]. Tento zámek zajišťuje, aby na jednom Matlab serveru nebylo vykonáváno více úloh současně. V dalším kroku jsou z tabulky `process` vybrána data (`processID, command`) nevyřešené úlohy. Pokud se nepodaří žádnou úlohu z databáze načíst, je zámek `ThreadLock` odemknut a Matlab serveru není přidělena žádná úloha. Pokud se získání úlohy podařilo jsou načteny parametry skriptů (název skriptu, modifikace) potřebných pro řešení dané úlohy. Získaná data jsou uložena do výstupní proměnné.

[12] Více o zámku ThreadLock v kap.4.6.3.



Obr. 4.31: SOAP server init

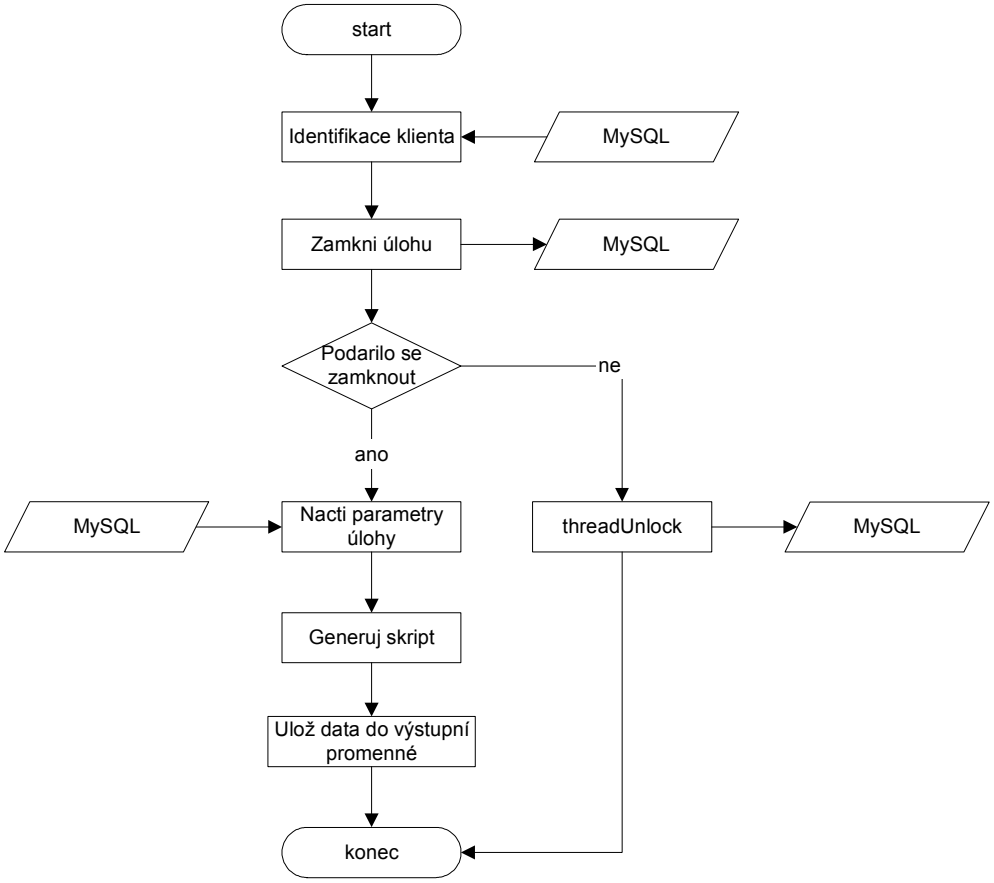
Loadscript

Služba pro poskytnutí skriptů uložených na SOAP serveru. Klient zažádá o poslání skriptů, které potřebuje pro řešení. Vstupem je seznam vyžadovaných skriptů. Výstupem je pole skriptů. Aby bylo posílání skriptů bezpečné je pro odesílané skripty vypočítán kontrolní součet pomocí hešovací funkce MD5.

Loaddata

Služba, která pro zadaný identifikátor úlohy vygeneruje kód pro její vyřešení. Diagram popisující tuto službu je na obr.4.32. Vstupem je identifikátor úlohy a výstupem kód pro výpočet rozvrhu pomocí algoritmu z TORSCHE. Najprve je identifikován klient stejným způsobem jako v případě služby init. Před vybráním dat potřebných pro řešení úlohy je tato

úloha zamknuta[13], aby nemohla být řešena jiným serverem. Pokud je zamčení úlohy neúspěšné, je odemčen zámek `threadLock` blokující řešení další úlohy a skript je ukončen s hodnotou false. Z dat, která jsou načtena z databáze je vytvořen kód spustitelný v MATLABu, který řeší danou úlohu. Generování kódu je zajištěno funkcí, která je podrobněji popsána v kapitole 4.6.2. Výsledný kód je odeslán Matlab serveru k vyřešení.



Obr. 4.32: SOAP server loaddata

Senddata

Prostřednicvím této služby Matlab server uloží výsledná data, která jí posílá Matlab server, do databáze a odemkne zámek, který blokoval řešení další úlohy na tomto Matlab serveru.

Serializetest

Služba pro ověření správné interpretace posílaných dat. Tato služba otestuje zda se dají serializovaná data poslaná klientem deserializovat. Vstupem jsou serializovaná data, výstupem je `true` nebo `false`, podle toho jestli se podařilo data deserializovat pomocí funkce unserialize.

[13] Více v kapitole 4.6.3.

4.6.2. Generování kódu

Generování kódu pro MATLAB je klíčovou součástí aplikace. Kód je generovaný při volání webové služby `loaddata`. Nejprve jsou z databáze načtena data o úloze z tabulek *`processSyntax`* a `process`. Tato data definují syntaxi dané úlohy a také odkazují na data uložená v dalších tabulkách. Z těchto dat je vytvořen kód pro vytvoření vstupních parametrů zvolené rozvrhovací funkce a pro její inicializaci. Kód 4.1. ukazuje generování kódu pro vytvoření sady úloh. Dalším krokem je přidání kódu pro zpracování výsledků, které vrátí rozvrhovací funkce. Protože je výstup funkcí TORSCHE standardizovaný je tento kód stejný pro všechny úlohy a pouze je připojen na konec kódu vygenerovaného v předchozím kroku.

```
$taskset = "set = taskset(["";
while(list($tid,$var,$proc,$rel,$dead,$due,$w,$p) = mysql_fetch_array($result)){
        $taskOrd["."$tid."]=$i;
        $task = 'task(';
        $task .= empty($var)?"":"""."$var."","";

        $task .= $proc;
        if(!empty($rel)){$task .= ","."$rel;
                if(!empty($dead)){$task .= ","."$dead;
                        if(!empty($due)){$task .= ","."$due;
                                if(!empty($w)){$task .= ","."$w;
                                        if(!empty($p)){$task .= ","."$p;}}}}}
        $task .= ")";
        $taskset .= " "."$task;
        $i++;}
```

kód. 4.1: generování kódu pro vytvoření sady úloh

4.6.3. Systém zámků

Bezpečné přidělování úloh SOAP serverem je zajištěno systémem zámků. Systém využívá pro svou funkci databázi MySQL a v ní implementované unkátní indexy v tabulkách. Unikátní index v tabulce aplikovaný na nějaký sloupec znemožňuje zapsat do tohoto sloupce hodnoty, které již jsou v něm uložené. Tedy pokud se pokusí nějaká aplikace do sloupce s unikátním indexem zapsat hodnotu v něm již uloženou je tato operace se nevykoná.
V aplikaci se používají zámky, aby se zabránilo řešení více úloh na jednom serveru a aby nedocházelo k řešení stejné úlohy na více serverech. V databázi jsou pro zámky použity tabulky `solver`, `schedule` a `threadLock`. Pokud se nějaký výpočetní prostředek pokouší vzít úlohu, musí se nejprve pokusit zapsat do tabulky `thredLock`. V tabulce `threadLock` jsou uvedeny identifikátory jednotlivých výpočetních prostředků, které řeší nějakou úlohu. Pokud tedy je na daném výpočetním prostředku již nějaká úloha vykonávána, je identifikátor tohoto výpočetního prostředku zapsaný v tabulce `threadLock` a unikátní index zabrání vykonání dašího zápisu stejného identifikátoru. Pokud nějaký výpočetní prostředek chce začít řešit

nějakou úlohu, musí se pokusit zapsat do tabulky `schedule`, kde jsou uvedeny již vyřešené nebo právě řešené úlohy. Unikátní index v tabulce `schedule` na sloupci identifikátoru úlohy zabraňuje zapsání tohoto identifikátoru vícekrát. Pokud se tedy zapsání do tabulky nepovede je daná úloha zahozena a výpočetní prostředek se pokusí vybrat jinou úlohu.

4.6.4. Modifikace

SOAP server byl navržen tak, aby se dal jednoduše modifikovat pro práci s jiným typem klienta a jinými výpočetními skripty. Modifikace nastavení SOAP serveru je založena na systému přídavných skriptů, které jsou připojeny k dané webové službě. Webové služby jsou uloženy v souboru `ServerServices.php`. Každá webová služba je rozdělena příkazem switch podle hodnoty vstupního parametru `interpret`. Tyto části určují, jaký skript má být pro danou hodnotu proměnné `interpret` vykonáván. Pokud chce uživatel přidat další typ výpočetního prostředku, přidá do příkazu `switch` další hodnotu, kterou se bude prostředek identifikovat v proměnné `interpret` a přidá skript, který má daný prostředek obsloužit. Kód 4.2. ukazuje přiřazování skriptů pro obsluhu různých typů výpočetních prostředků.

```
function init($sdata){
        $data = unserialize($sdata);
        switch($data['interpret']){
                case 'matlab':
                        include 'matlab/init.inc.php';
                        return isset($return)?serialize($return):false;
                        break;
                case 'c':
                        include 'c/init.inc.php';
                        return isset($return)?serialize($return):false;
                        break;
                case 'c++':
                        include 'cpp/init.inc.php';
                        return isset($return)?serialize($return):false;
                        break;}}
```

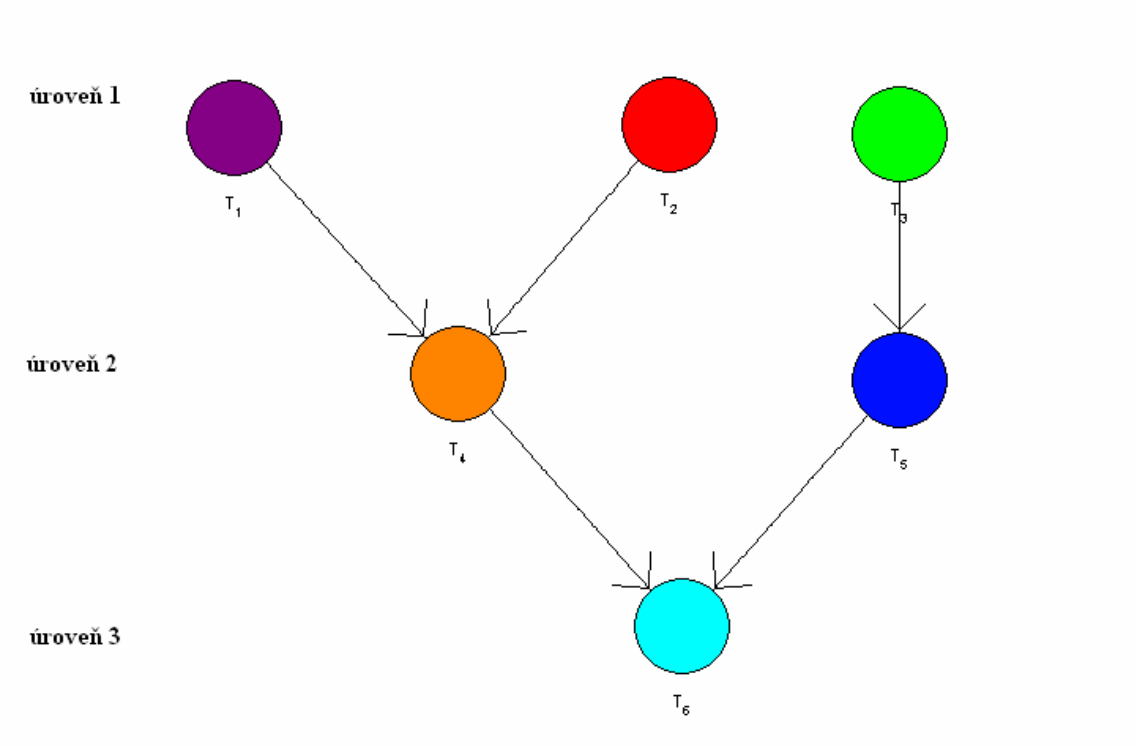
kód 4.2: rozlišení typu výpočetního prostředku

5. Implementované algoritmy

Pro rozšíření scheduling toolboxu a pro otestování funkčnosti rozhranní jsem implementoval dva rozvrhovací algoritmy. Jedná se o algoritmy, které řeší minimalizaci celkové délky rozvrhu Cmax pro úlohy s precedenčním omezením, které jsou řešené na identických procesorech.

5.1. Huův algoritmus

Tento algoritmus je navržen pro řešení problému P|in-tree,pj=1|Cmax. Úlohy jednotkové délky jsou vykonávány na identických procesorech. Precedenční omezení jsou ve tvaru in-tree. V algoritmu je použito značení úrovní in-tree, kde úroveň značí počet úloh na cestě ke kořenu stromu obr. 5.1. Algoritmus může být použit také pro řešní úlohy s precedenčními závislostmi ve tvaru out-tree. V tom případě jsou precedenční závislosti zadány s opačnou orientací a výsledný rozvrh čteme pozpátku. Pokud jsou precedenční závislosti složeny z více in-tree(out-tree), je možné tento algoritmus použít také pokud přidáme falešnou úlohu, která je následníkem všech kořenů in-tree, tím vytvoříme celkovou in-tree(out-tree) strukturu.



Obr. 5.1: Úrovně in-tree

5.1.1. Algoritmus

```
Begin;

vypočítej úrovně úloh v in-tree;

t=0;

repeat

      vytvoř list Lt úloh, které v čase t nemají předchůdce;

         - buď nemá předchůdce, nebo předchůdci jsou vykonáni

      seřaď úlohy Lt v nerostoucím pořadí jejich úrovní;

     přiřaď m(jestli jich je m) úloh ze začátku listu Lt na procesory;

      vyjmi přiřazené úlohy z listu;

      t=t+1;

until všechny úlohy jsou rozvrhnuty;

end;
```

Je zřejmé, že časová závislost algoritmu *hu* na počtu rozvrhovaných úloh je lineární. Časová složitost je tedy *O(n)*.

5.1.2.Implementace

V prostředí MATLAB jsem algoritmus implementoval funkcí *hu*.

```
      TS = hu(T, p, m[,verbose])
      TS = hu(T, p, m[,schoptions])
```

Syntaxe použitá při volání algoritmu *hu* vyžaduje tři povinné vstupní proměnné, které mohou být doplněny jednou nepovinnou proměnnou. Povinné vstupní hodnoty jsou sada úloh obsahující precedenční omezení, kterou chceme rozvrhnout `T`, rozvrhovací problém `p` a počet processorů `m`. Nepovinné údaje `verbose` a `schoptions` určují zda se mají na standardní výstup vypisovat informace o průběhu rozvrhování. Z proměnné `schoptions` je čtena hodnota `verbose`, jejíž hodnotu definujeme pomocí funkce `schoptionsset`.

```
      Option = schoptions('verbose',1)
```

Proměnné `T` a `p` jsou objekty TORSCHE scheduling toolboxu vytvořené pomocí funkcí `taskset` a `problem`. Výstupem funkce *hu* je rozvrh vytvořený Huovým algoritmem.

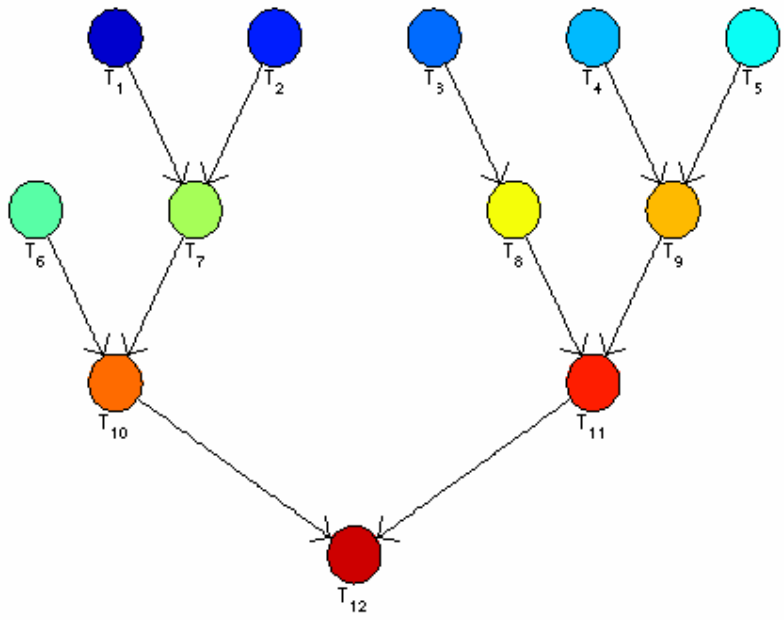
Vzhledem k tomu, že Huův algoritmus je využíván v algoritmu Coffman and Graham[14] na rozvržení ocedulkovaných úloh, byl algoritmus upraven, aby mohl být na vstupu zadán rozvrhovací problém P2|prec,pj=1|Cmax a sada úloh s přiřazenými cedulkami. Syntaxe volání

[14] Algoritmus popsán v kapitole 5.2.

funkce *hu* pro rozvržení ocedulkovaných úloh zůstává zachována s tím, že rozvrhovací problém je P2|prec,pj=1|Cmax, počet procesorů je 2 a jednotlivé úlohy ze sady úloh obsahují v proměnné `userParam` hodnotu přiřazené cedulky.

5.1.2.1. Příklad

Mějme sadu úloh jednotkové délky s precedenčními závislostmi dle obrázku obr. 5.2, které chceme rozvrhnout na tři procesory.

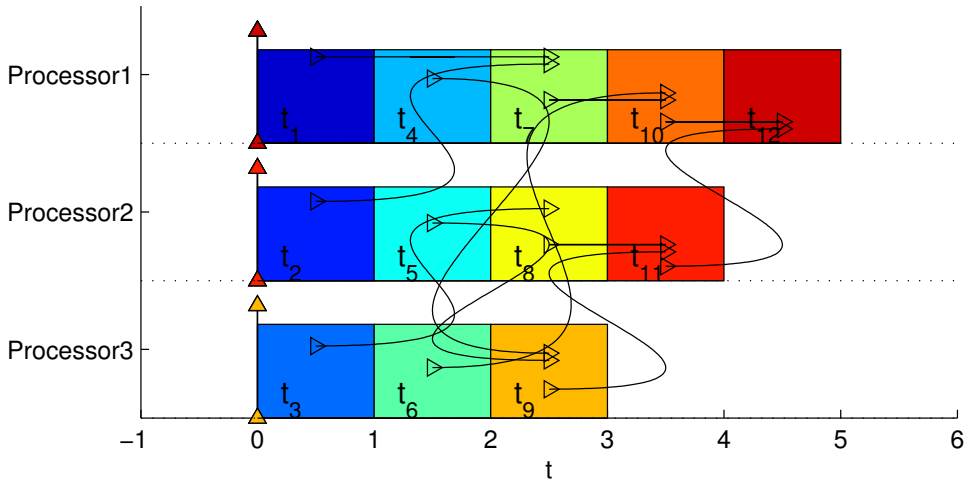


Obr. 5.2: Úloha pro Huův algoritmus – zadání

V prostředí MATLAB úlohu vyřešíme například zápisem kódu:

```
>> p = problem('P|in-tree,pj=1|Cmax');
>> prec = [
      0 0 0 0 0 0 1 0 0 0 0 0
      0 0 0 0 0 0 1 0 0 0 0 0
      0 0 0 0 0 0 0 1 0 0 0 0
      0 0 0 0 0 0 0 0 1 0 0 0
      0 0 0 0 0 0 0 0 1 0 0 0
      0 0 0 0 0 0 0 0 0 1 0 0
      0 0 0 0 0 0 0 0 0 1 0 0
      0 0 0 0 0 0 0 0 0 0 1 0
      0 0 0 0 0 0 0 0 0 1 0 0
      0 0 0 0 0 0 0 0 0 0 0 1
      0 0 0 0 0 0 0 0 0 0 0 1
      0 0 0 0 0 0 0 0 0 0 0 0
      ];
>> T = taskset([1 1 1 1 1 1 1 1 1 1 1 1],prec);
>> TS = hu(T,p,3);
>> plot(TS);
```

Výsledný rozvrh ve formě ganttova diagramu je na obr.5.3.



Obr. 5.3: Úloha pro Huův algoritmus – řešení

Z výsledného rozvrhu je partné, že všechny precedenční omezení jsou splněny a maximální délka rozvrhu je 5 jednotek. Příklad byl použit z literatury [Błażewicz].

5.2. Algorithmus Coffman and Graham

Algoritmus Coffman and Graham umožňuje řešení problému jednotkových úloh s precednčními závislostmi ve tvaru kombinací in-tree a out-tree na dvou identických procesorech. Rozvrhovací problém je dle standardní notace ve tvaru P2|prec,pj=1|Cmax. Algoritmus je založen na používání pojmu cedulka, která bere v úvahu jak úroveň úlohy, tak počet jejích okamžitých následníků.

5.2.1. Algoritmus

```
Begin;
přiřaď cedulku 1 jakékoli úloze, která nemá následníka;
j:=1;
repeat
    vytvoř množinu S z neocedulkovaných úloh, jejichž následník má cedulku;
    for all T ∈ S do
        begin
        vytvoř seznam L(T) složený z cedulek následovníků T;
        seřaď L(T) v klesajícím pořadí cedulek;
        end;
    seřaď seznamy v rostoucím lexografickém pořadí L(T[1])∠L(T[2])∠…∠L(T[|S|]);
    přiřaď cedulku j+1 úloze T[1];
    j:=j+1;
until j=n+1;
zavolej algoritmus Hu, místo úrovní použij cedulky;
end;
```

Algoritmus tedy pracuje ve dvou krocích. Nejprve jsou úlohám přiděleny cedulky, poté jsou úlohy rozvrženy pomocí Huova algoritmu, kde jsou cedulky použity místo úrovní. Huův algoritmus tedy musel být upraven, aby akceptoval také úlohy s předem určenými úrovněmi.

5.2.2. Implementace

Algotitmus jsem v prostředí MATLAB implementoval funkcí *coffmangraham*.

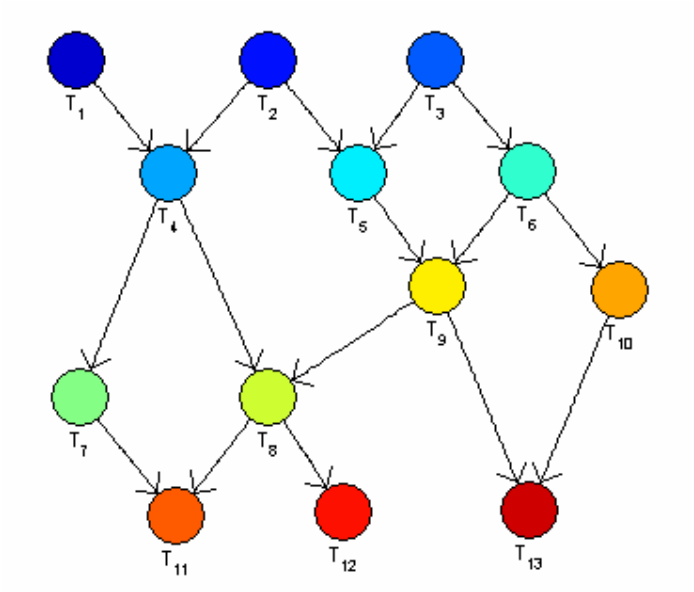
```
TS = coffmangraham(T, p[,verbose])
TS = coffmangraham(T, p[,schoptions])
```

Povinné vstupní údaje jsou sada úloh `T` a rozvrhovací problem `p`. Nepovinné údaje jsou stejně jako u funkce *hu* `verbose` a `schoptions`, které určují zda se mají na standardní výstup

vypisovat informace o průběhu rozvrhování. Stejně jako tomu bylo u funkce *hu* v proměnné `schoptions` nastavujeme hodnotu `verbose`.

5.2.2.1. Příklad

Mějme sadu třinácti úloh jednotkové délky s precedenčním omezením dle obrázku obr. 5.4.



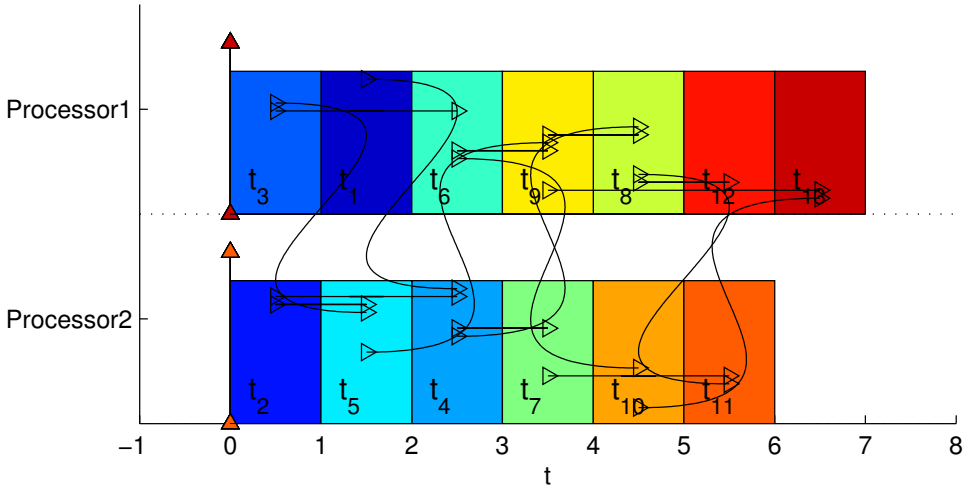
Obr. 5.4: Zadání pro algoritmus Coffman and Graham

Takto definované precedenční omezení v MATLABu vytvoříme ve tvaru matice `prec`.

```
>> prec = [
      0 0 0 1 0 0 0 0 0 0 0 0 0
      0 0 0 1 1 0 0 0 0 0 0 0 0
      0 0 0 0 1 1 0 0 0 0 0 0 0
      0 0 0 0 0 0 1 1 0 0 0 0 0
      0 0 0 0 0 0 0 0 1 0 0 0 0
      0 0 0 0 0 0 0 0 1 1 0 0 0
      0 0 0 0 0 0 0 0 0 0 1 0 0
      0 0 0 0 0 0 0 0 0 0 1 1 0
      0 0 0 0 0 0 0 1 0 0 0 0 1
      0 0 0 0 0 0 0 0 0 0 0 0 1
      0 0 0 0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0 0 0 0
      ];
```

Vytvoříme sadu úloh `T` a definujeme problém `p`.Úlohy poté rozvrhneme pomocí funkce *coffmangraham*. Výsledný rozvrh ve tvaru ganttova diagramu je na obrázku obr. 5.5.

```
>> T = taskset([1 1 1 1 1 1 1 1 1 1 1 1 1],prec);
>> p = problem('P2|prec,pj=1|Cmax');
>> TS = coffmangraham(T, p);
>> plot(TS);
```



Obr. 5.5: Algorimus Coffman and Graham – řešení

Příklad byl použit z literatury [Błażewicz].

6. Testy vzdáleného rozvrhování

Pro ověření správné funkčnosti jednotlivých součástí aplikace jsem provedl sadu testů.

6.1. Test zámků

Zámky v aplikaci slouží pro zamezení konfliktů při výběru úloh a pro omezení počtu vláken nebo procesů řešících úlohy na Matlab serverech. Při testování možných konfliktů nás nezajímá vykonávání úlohy samotné, ale pouze okamžik jejího přidělení Matlab serveru a uložení výsledků. Z toho důvodu je při testu použito velké množství úloh s krátkou dobou vykonávání. Pro otestování zámku vláken a procesů na serverech jsou servery při vykonávání úloh několikrát znovu inicializovány.

6.1.1. Zadání testu

Pro účel testování bylo zadáno tisíc jednoduchých úloh, jejichž průměrná doba vykonávání na použitých Matlab severech byla 0,25 s. Pro rozvrhování byl použit algoritmus List scheduling řešící problém `P|prec|Cmax` s parametry úloh dle tabulky 6.1 a precedenčními závislostmi dle obrázku 6.1.

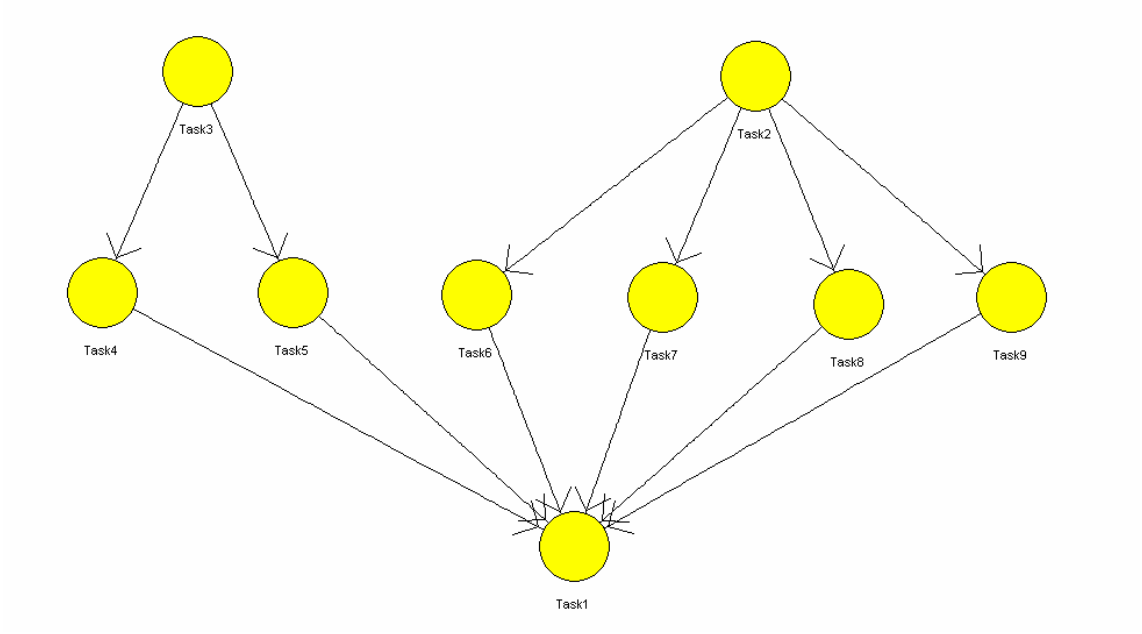
Úloha	Doba vykonávání	Okamžik disponibility
Task1	5	0
Task2	10	0
Task3	10	0
Task4	15	0
Task5	15	0
Task6	15	0
Task7	15	0
Task8	15	0
Task9	20	0

Tabulka 6.1: zadání testu – úlohy

Úlohy byly řešeny na třech Matlab serverech s konfigurací dle tabulky 6.2. V průběhu řešení byly servery třikrát znovu inicializovány pro otestování zámku vláken a procesů.

Server	Inicializace	MATLAB	OS	Procesor	RAM [MB]
1	Matlab Web Server	7.1.0.246(R14)	MS Windowx XP	Intel P4	2046
2	Remote run CGI	7.1.0.246(R14)	MS Windows XP	Intel P4	512
3	Remote run CGI	7.0.4.352(R14)	Debian Linux	Intel P3	2046

Tabulka 6.2: zadáni testu - Matlab servery



Obr. 6.1: zadání testu - precedenční závislosti

6.1.2. Výsledky testu

Informace o průběhu řešení úloh a použití zámků jsou ukládány v databázi do tabulky `log`. Pokud došlo při přidělování úloh ke konfliktu a jeho vyřešení pomocí zámků, byla tato skutečnost zaznamenína v tabulce `log` takto:

```
2007-05-19 18:36:25 Machine 2 assigned process 52, thread locked
2007-05-19 18:36:25 Machine 1 assigned process 52, thread locked
2007-05-19 18:36:25 Machine 2 process 52 locked
2007-05-19 18:36:25 Machine 1 process 52 refused
```

Zámky spuštění dalšího vlákna nebo procesu na Matlab serveru, byly v tabulce log zaznamenány takto:

```
2007-05-19 18:36:54 Machine 2 thread refused
```

Celkový počet zaznamenaných a vyřešených kolizí přidělování úloh pro jednotlivé Matlab servery je uveden v tabulce 6.3. Z těchto dat je patrné, že zámky úloh fungují správně a došlo k ošetření všech vzniklých konfliktů.

Server	Řešeno úloh	Úspěšně vyřešeno	Počet kolizí	Ošetřeno kolizí
1	462	462	49	49
2	187	187	29	29
3	351	351	41	41
Sum	**1000**	**1000**	**119**	**119**

Tabulka 6.3: výsledky testu – kolize

Všechny pokusy o spuštění dalších vláken, nebo procesů byly ošetřeny pomocí zámku vláken.

6.2. Test rychlosti

Cílem tohoto testu je zjistit efektivitu paralelního zpracování kap.4.2.2.

6.2.1. Zadání testu

Testy byly prováděny na Matlab serverech s konfigurací dle tabulky 6.2. Zadání testovaných úloh odpovídá zadání z kap.6.1.1. Pro zadaný počet úloh je měřen časový interval mezi časem odeslání úlohy k vyřešení na Matlab serveru a časem uložení výsledku poslední řešené úlohy do databáze. Tímto zjistíme jak dlouho by musel uživatel, který úlohy zadal čekat na výsledky úloh.

6.2.2. Výsledky testu

Nejdříve byly změřeny, pro porovnání, závislosti doby výpočtu na počtu úloh pro jednotlivé Matlab servery. Tato data jsou v tabulce 6.4. Z nich je patrné, že závislosti jsou přibližně lineární. U každého ze serverů je patrné zpoždění výpočtů, které je způsobeno inicializací. V tabulce 6.5 jsou vypsány změřené hodnoty zpoždění pro jednotlivé Matlab servery. Z těchto dat je patrné, že nejkratší zpoždění bylo zjištěno u serveru, který je inicializován prostřednictvím MATLAB Web Serveru. Toto je dané tím, že u ostatních Matlab serverů je při inicializaci spuštěn program MATLAB, zatímco u MATLAB Web Serveru je již MATLAB spuštěný a inicializují se jen skripty potřebné pro výpočet.

Server	počet úloh n[]	10	50	100	500	1000
1	t[s]	17	83	166	823	1644
2	t[s]	66	240	427	2356	4624
3	t[s]	36	121	228	1072	2139

Tabulka 6.4: Výsledky testu rychlosti pro jednotlivé Matlab servery

Server	zpoždění t[s]
1	0
2	15
3	8

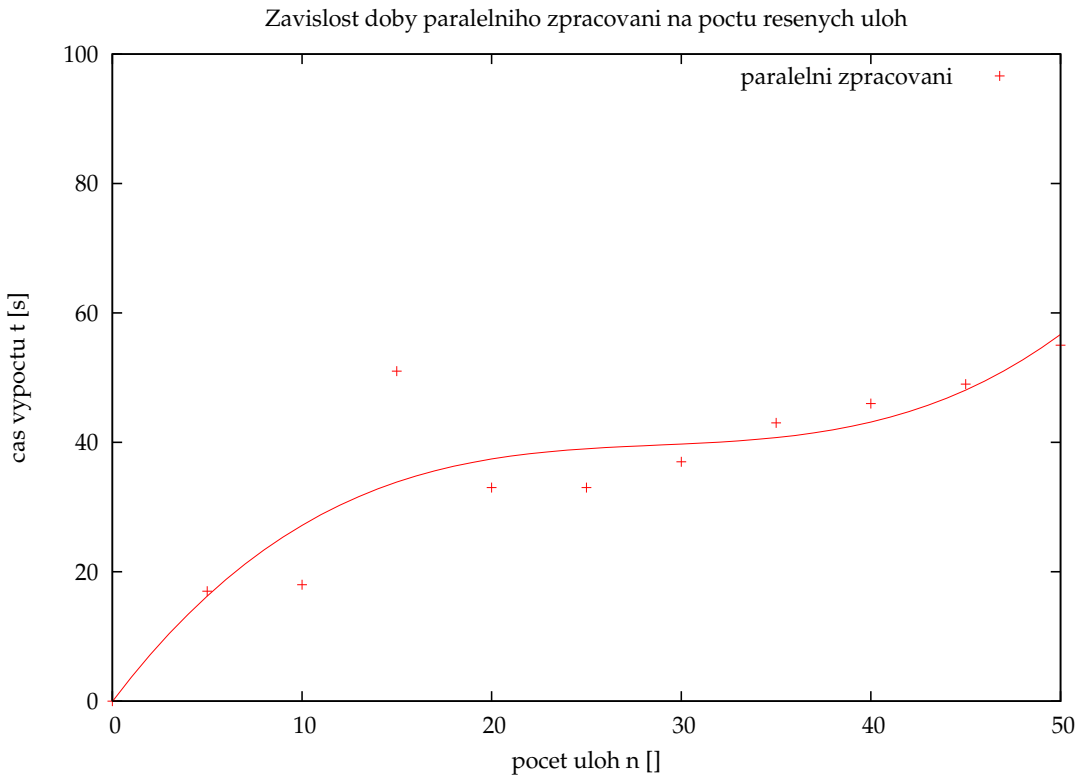
Tabulka 6.5: Zpoždění daná inicializací

Dále byla změřena závislost doby výpočtu pomocí paralelního zpracování (kap.4.2.2) na počtu zadaných úloh. Naměřeny hodnoty jsou v tabulce 6.5. Z těchto dat je patrné že závislost není z počátku lineární, ale až pro řešení většího počtu úloh ji můžeme považovat za lineární. Průběh této nelinearní závislosti je zobrazen na obr.6.2. Pro malý počet úloh je řešení prováděno pouze na MATLAB Web Serveru a ostatní Matlab servery se vlivem zpoždění k řešení nepřipojují. V okamžiku, kdy se k řešení připojí Matlab server 2, dojde vzhledem

k jeho menší výpočetní kapacitě k prodloužení celkové doby řešení[15]. Při řešení většího množství úloh je vliv menší výpočetní kapacity serveru potlačen a závislost je přibližně lineární. Grafu na obrázku obr.6.3 ukazuje porovnání dob výpočtů jednotlivých Malab serverů a výpočtů pomocí paralelního zpracování. Z tohoto grafu je patrné, že paralelní zpracování výrazně zrychluje řešení úloh.

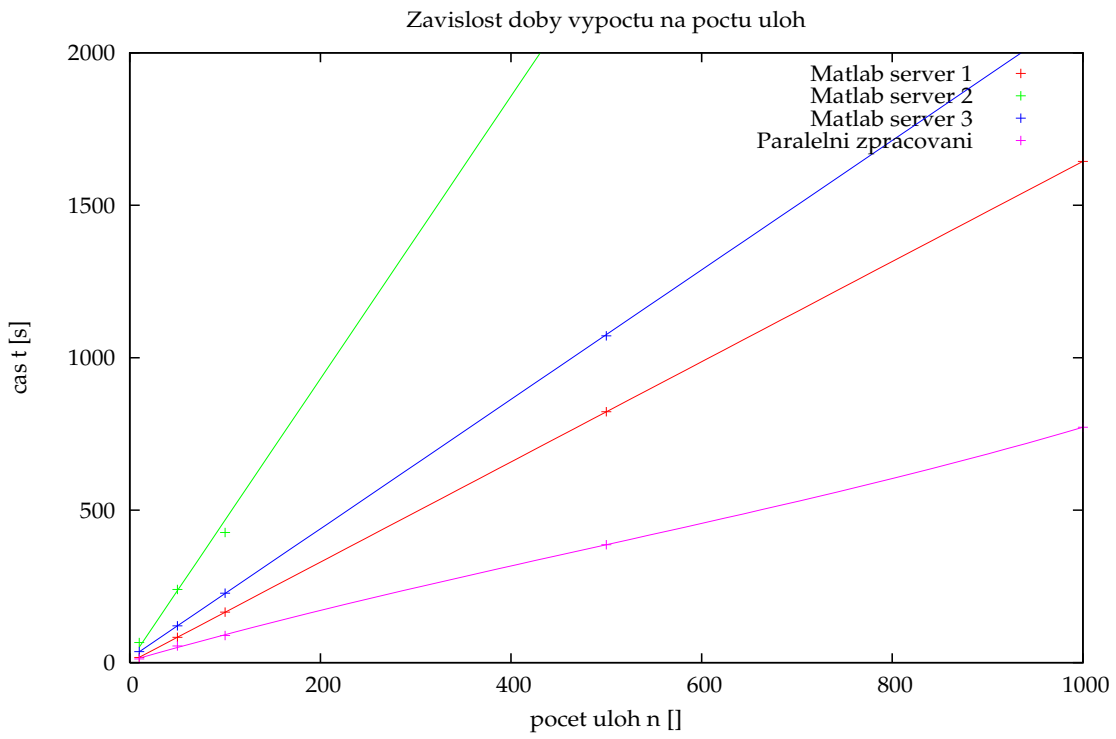
počet úloh n[]	5	10	15	20	25	30	35	40	45	50	100	500	1000
t[s]	17	18	50	32	32	37	43	46	49	55	89	397	772

Tabulka 6.5: půběh paralelního zpracování



Obr. 6.2: počáteční nelinearita paralelního zpracování

[15] Pokud bychom použili pro testování servery se shodnou výpočetní kapacitou k tomuto jevu by nedošlo.



Obr. 6.3: graf závislosti doby vykonávání na

6.3. Test rozhranní

Rozhranní bylo testováno za použití webových prohlížečů Internet Explorer 7, Mozilla Firefox 1.5.0.11, Opera 7.54u2 a Konqueror 3.5.5. Cílem testu bylo zjistit zda je rozhranní zobrazeno pomocí těchto prohlížečů stejně a nabízí všechny nezbytné funkce impementované na rozhranní.

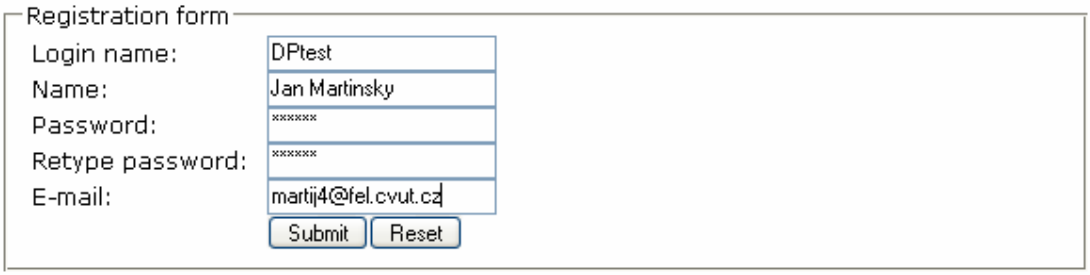
6.3.2. Výsledky testu

Při práci s rozhranním pomocí prohlížečů Internet Explorer 7, Mozilla Firefox a Konqueror se nevyskytly žádné chyby funkčnosti ani zobrazení webových stránek. Při použití prohlížeče Opera 7.54u2 došlo k nepřesnému zobrazení menu sady úloh v aplikaci, funkčnost však zůstala zachována. Tento problém se mi nepodařilo odstranit.

7. Demonstrace použití vzdáleného rozvrhování

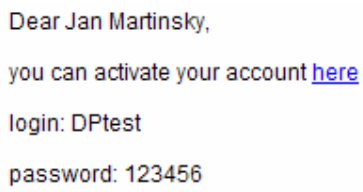
V této kapitole je uveden příklad použití rozhranní pro vzdálené rozvrhování. Pro demonstraci je použit algorithmus Coffman and Graham. Zadání úlohy odpovídá zadání z kapitoly 5.2.2.1.

Nejdříve se zaregistrujeme na stránce registration.php obr.7.1.



Obr. 7.1: registrace

Obratem obdržíme e-mail s našemi přihlašovacími údaji a odkazem na webovou stránku, kde můžeme účet aktivovat obr.7.2.

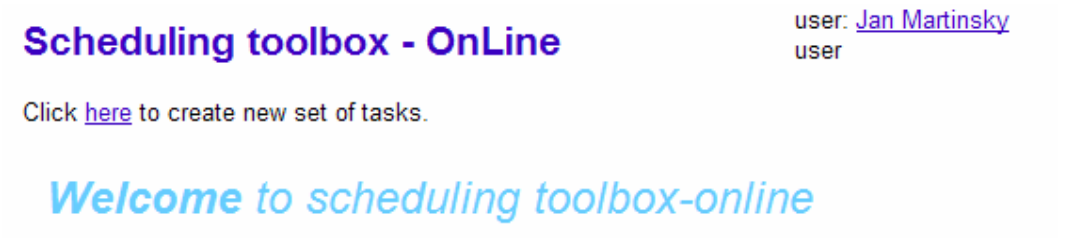


Obr. 7.2: aktivace účtu

Kliknutím na odkaz se náš účet aktivuje a zobrazí se nám hlavní stránka s formulářem pro přihlášení obr.7.3.

Obr. 7.3: přihlášení

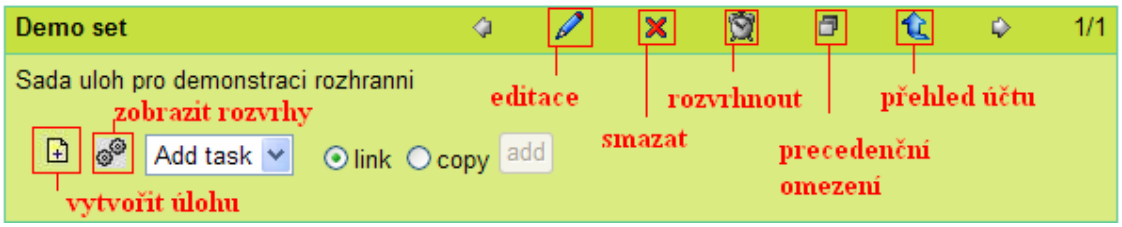
Po přihlášení je zobrazena stránka s přehledem našeho účtu a odkaz na formulář pro zadání úlohy obr.7.4. Protože náš účet zatím nic neobsahuje je stránka prázdná. Kliknutím na odkaz se zobrazí formulář do kterého vyplníme jméno, jméno proměnné a popis vytvářené sady úloh obr.7.5.



Obr. 7.4: vstupni stranka

Obr. 7.5: vytvoření nové sady úloh

Odesláním vyplněného formuláře se dostaneme na stránku vytvořené sady úloh. Vzhledem k tomu, že sada ještě neobsahuje žádnou úlohu, je v menu patrné, že možnosti rozvrhování a nastavení precedenčních omezení jsou blokovány (šedé ikonky). Také zobrazení rozvrhu není umožněno. V tuto chvíli můžeme pouze editovat parametry sady úloh, smazat sadu úloh, vytvořit nebo přidat úlohy, nebo se vrátit na stránku s přehledem účtu.Pro vytvoření nových úloh klikneme na ikonku pro vytvoření úlohy. Zobrazí se nám stránka s formulářem obr.7.7.

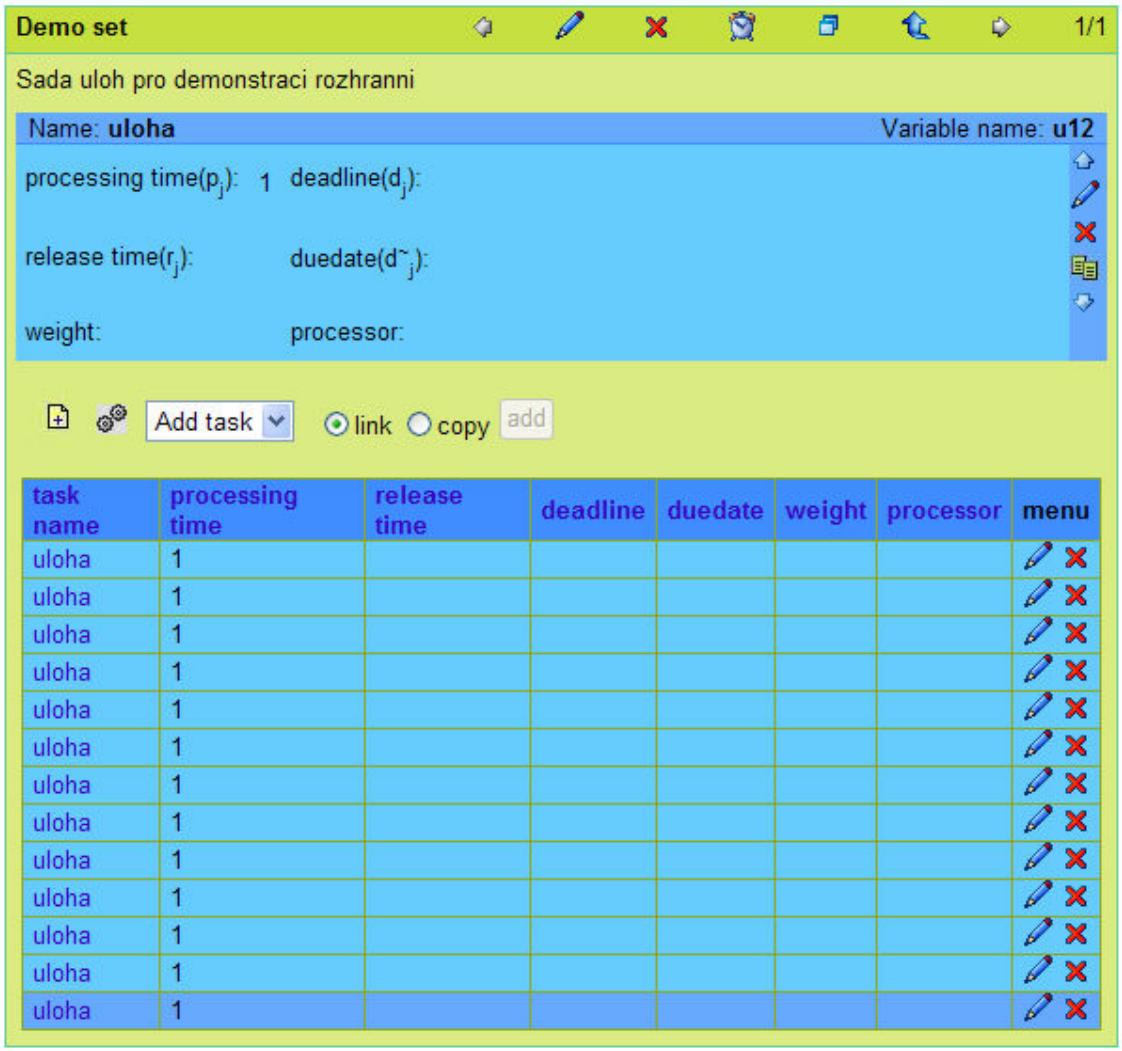


Obr. 7.6: nová sada úloh

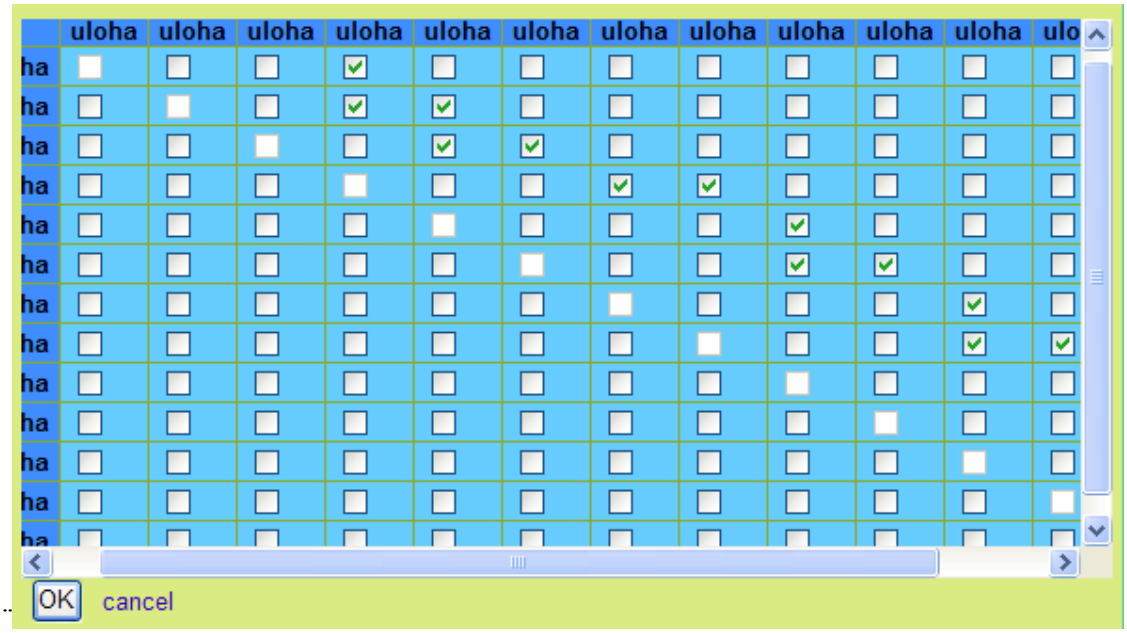
Protože, dle zadání z kap.5.2.2.1 potřebujeme vytvořit 13 úloh jednotkové délky, vyplníme pouze kolonky formuláře s názvem úlohy, názvem proměnné a dobou vykonávání. Parametr copy, který určuje kolik kopií dané úlohy se má vyvořit, nastavíme na hodnotu 13.

Obr. 7.7: vytvoření úloh

Po odeslání formuláře dojde k vytvoření 13 úloh a ty se zobrazí viz obr.7.8. v tuto chvíli už je umožněno rozvrhování, ale nejprve nastavíme precedenční omezení obr.7.9.

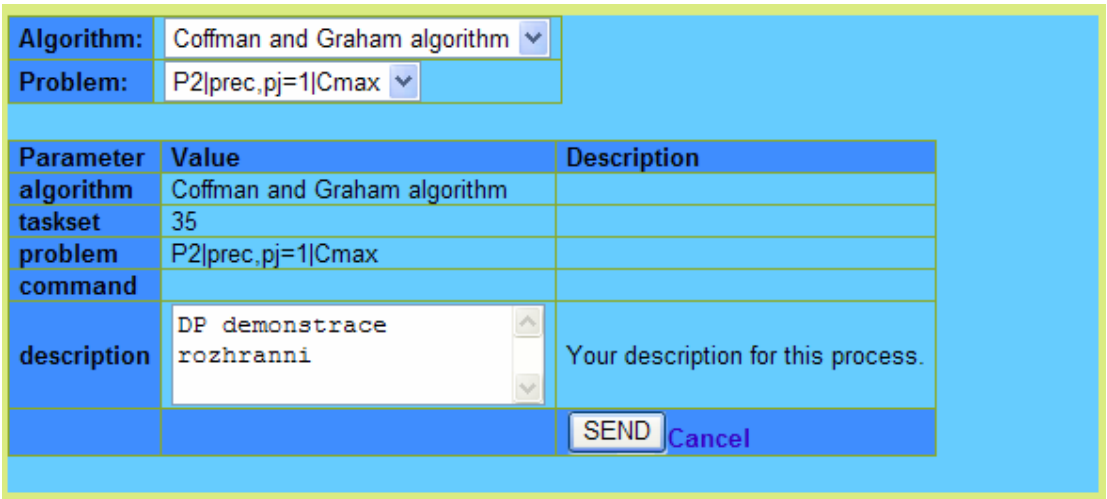


Obr. 7.8: vytvořená sada úloh s úlohami



Obr. 7.9: nastavení precedenčních omezení

Po nastavení precedenčních omezení přejdeme na odkaz, který ná umožní vybrat algoritmus a problém, který chceme řešit obr.7.10. Zvolíme algritmus Coffman and Graham a dále můžeme jen zvolit rozvrhovací problém P2|prec,pj=1|Cmax. Vyplníme popis a odešlme.



Obr. 7.10: zadání zvoleného algoritmu a problému

Nyní je námi zadaná úloh řešena na některém z Matlab serverů. Klikneme na ikonu, pro zobrazení rozvrhu a zobrazí se nám výsledný rozvrh námi zadané úlohy obr.7.11 a gantů diagram vytvořený v MATLABu ve formátu PNG obr.7.12. Z výsledného rozvrhu jsou patrné hodnoty kritérií a jak jsou jednotlivé úlohy přidělovány na procesoty. Kliknutím na odkaz `Task Schedule` si můžeme prohlédnout rozvrh a jeho parametry ve formátu XML.

Demo set 1/1

Filter: processes all set of tasks Demo set OK

Process details status: **solved**

DP demonstrace rozhranni

Algorithm	Coffman and Graham algorithm
Problem	P2|prec,pj=1|Cmax

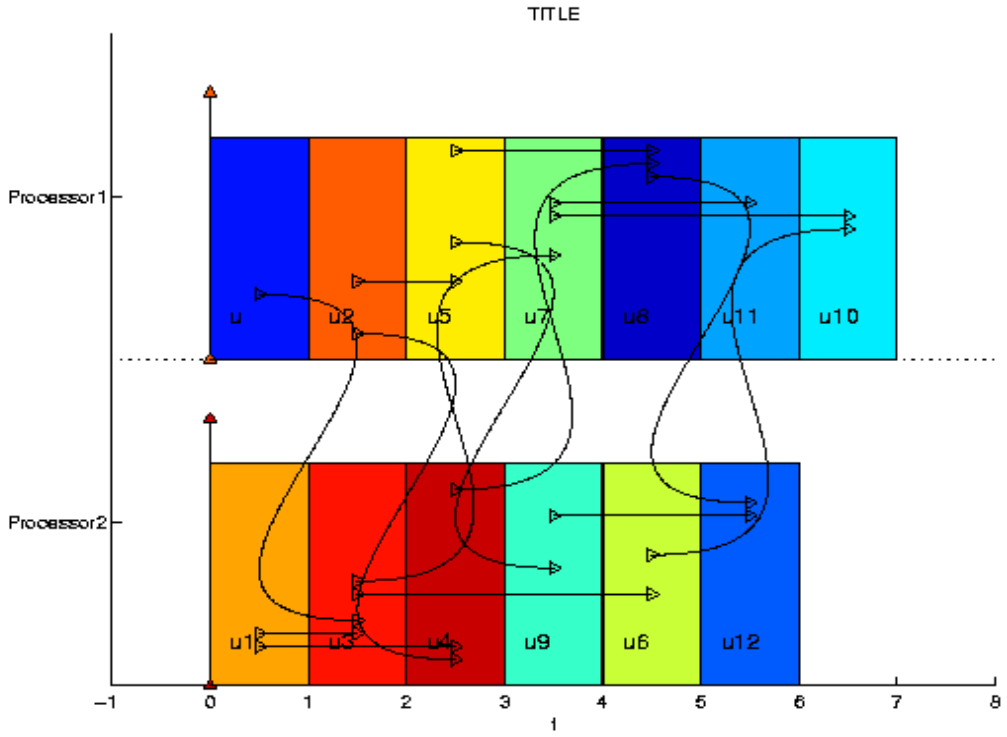
Task Schedule

name	start	length	processor
uloha	4	1	1
uloha	0	1	1
uloha	5	1	2
uloha	5	1	1
uloha	6	1	1
uloha	3	1	2
uloha	3	1	1
uloha	4	1	2
uloha	2	1	1
uloha	0	1	2
uloha	1	1	1
uloha	1	1	2
uloha	2	1	2

Schedule param

name	value
cmax	7
sumcj	49
sumwcj	49
time	0.1

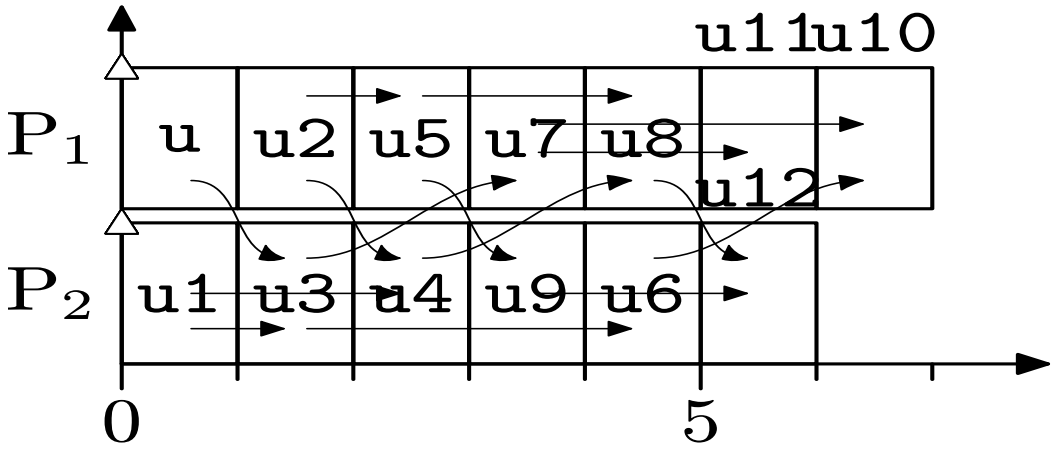
Obr. 7.11: rozvrh úlohy



Obr. 7.12: ganttův diagram

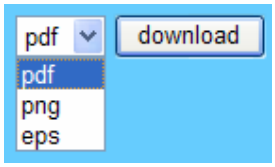
Pokud se nám ganttův diagram nelíbí, můžeme si pomocí nástroje PsGantt vygenerovat jiný. Pokud chceme nahrajeme do databáze soubor s kaskádovovými styly pro generováni ganttova diagramu. Poté pomocí menu PsGantt obr.7.13, necháme vygenerovat ganttův diagram obr.7.14.

Obr. 7.13: generování ganttova diagramu (PsGantt)



Obr. 7.14: ganttův diagram vytvořený pomocí PsGantt

Pokud si chceme vygenerovaný ganttův diagram stáhnout na počítač použijeme menu obr.7.15, kde nastavíme v jakém formátu chceme obrázek uložit.



Obr. 7.15: uložení diagramu na disk

Rozhranní dále obsahuje doplňkové funkce pro usnadnění práce a obsluhu uživatelského účtu.

8. Závěr

Cílem této práce bylo navrhnout a implementovat webové rozhranní, které umožní vzdálené rozvrhování s využitím TORSCHE Scheduling Toolboxu pro Matlab. Webové rozhranní bylo implementováno pomocí technologie PHP a JavaScript na webovém serveru. Pro načítání dílčích částí stránek jsem použil konceptu AJAX. Webové stránky splňují standard XHTML a byly úspěšně validovány pomocí W3C Markup Validation Service. Rozvrhování je prováděno na Matlab serverech s využitím TORSCHE Scheduling Toolboxu pro Matlab. Matlabovský skript použitý pro řešení úloh rozvrhování je nezávislý na řešených úlohách a je možné ho použít pro řešení jiného typu úloh. Aby byla odezva na zadání úlohy, co nejrychlejší, je umožněn výpočet více úloh najednou paralelně na několika Matlab serverech. Pro uchovávání dat je použita relační databáze MySQL. Původním záměrem bylo použít pro komunikaci s webovým rozhranním pouze aplikaci MATLAB Web Serveru. Vzhledem k tomu, že podpora a vývoj této aplikace je zastaven, navrh jsem vlastní aplikaci umožňující vzdálené spouštění skriptu pro Matlab. Výhodou této aplikace je, že výpočetní prostředí MATLAB nemusí být na serveru neustále spuštěno, ale je spouštěno prostřednictvím CGI skriptu až v případě, kdy je potřeba řešit nějaké úlohy. Nevýhodou naopak je časové zdržení výpočtů způsobené nabíháním MATLABu. Toto zdržení však ovlivňuje pouze řešení první úlohy, kdy dojde ke spuštění MATLABu, poté je MATLAB aktivní dokud jsou mu přidělovány úlohy z databáze. Komunikace Matlab serveru s databází byla původně implementována pomocí mex-souboru [MATLAB], který umožňoval rychlý přenos dat. Od použití tohoto souboru bylo nakonec upuštěno, kvůli jeho nepřenositelnosti mezi různými platformami. Pro komunikaci bylo nakonec využito webových služeb, které poskytuje SOAP server implementovaný v PHP. SOAP server jsem navrhnul tak, aby umožňoval, kromě výpočetního prostředí MATLAB, použití i jiných výpočetních prostředků pro rozvrhování. Celkově aplikace funguje dle očekávání a je možné ji rozšířit o další nástroje.

Dále byly implementovány rozvrhovací algoritmy Coffman and Graham a Huův algoritmus, které řeší problémy P2|prec,pj=1|Cmax a P|in-tree,pj=1|Cmax. Huův algoritmus je implementován tak, aby mohl být použit pro rozvrhování sady ocedulkovaných úloh generované algoritmem Coffman and Graham. Navržené algoritmy byly přidány do TORSCHE Scheduling Toolboxu pro Matlab.

Literatura

[AJAX]

VRÁNA, J. *AJAX* [online]. 3.10.2005, [cit. 2007-05-14]. Dostupné z: <http://www.root.cz/clanky/ajax/> ISSN 1212-8309

[Błażewicz]

BLAZEWICZłażevicz, J., aj. *Scheduling Computer and Manufacturing process*, 2. vyd., Springer, 2001.

[CSS]

BOS, B. *Cascading Style Sheets* [online]. 1999, poslední revize 22.5.2007 [cit. 22.5.2007]. Dostupné z: <http://www.w3.org/Style/CSS/>

[Černý]

ČERNÝ, R. *Vzdálené generování Ganttových diagramů.* Praha, 2006. Bakalářská práce na ČVUT. Fakulta elektrotechnická. Vedoucí bakalářské práce Ing. Michal Kutil.

[Hanzálek]

HANZÁLEK, Z. *Přednášky k předmětu 35RDU*, Praha: ČVUT-FEL, 2005.

[HTML]

RAGGETT, D., aj. *HTML 4.01 Specification* [online]. 24.12.1999, [cit. 2007-05-15]. Dostupné z: < http://www.w3.org/TR/html4/>

[Interval]

JÍCHA, R. *Salted hash – další krok ke zvýšení bezpečnosti* [online]. 10.2.2005, [cit. 2007-05-15]. Dostupné z: <http://interval.cz/clanky/salted-hash-dalsi-krok-ke-zvyseni-bezpecnosti/> ISSN 1212-8651

[JavaScript]

RIŠO, M. *JavaScript –1—Úvod* [online], 8.7. 2004, [cit. 2007-05-15]. Dostupné z: <http://www.linuxsoft.cz/article.php?id_article=237> ISSN 1801-3805.

[Karolík]

KAROLÍK, A. *Nástroj na kreslení Ganttových diagramů s využitím Metapostu.* Praha, 2006. Diplomová práce na ČVUT. Fakulta elektrotechnická. Vedoucí diplomové práce Ing. Michal Kutil.

[Mařík]

MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J. a kol. *Umělá inteligence (3)*, 1. vyd., Praha: Academia, 2001. ISBN 80-200-0472-6

[Mathworks]

THE MATHWORKS, INC. *The mathworks – MATLAB Web Server* [online]. c1994-2006, [cit. 2007-05-10]. Dostupné z: <http://www.mathworks.com/access/helpdesk_r13/help/toolbox/webserver/webserver.html>

[MATLAB]

THE MATHWORKS, INC. *MATLAB The Language of Technical Computing: External interfaces (Version 6)*, 4. vyd., Mathworks, Inc., 2000.

[PHP]

WELLING, L., THOMSON, L. *PHP a MySQL rozvoj webových aplikací*. Přel. RNDr. Jan Pokorný. 2. vyd., Praha: SoftPress, 2004.

[SOAP]

BOX, D., aj. *SOAP* [online]. 18.4.2000, [cit. 2007-05-22]. Dostupné z: <http://static.userland.com/xmlRpcCom/soap/SOAPv11.htm>

[SQL]

REFSNES DATA. *SQL Introduction* [online]. c1997-2007, [cit. 2007-05-16]. Dostupné z: <http://www.w3schools.com/sql/sql_intro.asp>

[Stibor]

STIBOR, M. *Optimální řízení nákladních výtahů*. Praha, 2006.Bakalářská práce na ČVUT. Fakulta elektrotechnická . Vedoucí bakalářské práce Ing. Michal Kutil.

[TORSCHE]

KUTIL, M. *TORCHE Scheduling Toolbox for Matlab* [online]. c2004, [cit. 2007-05-10]. Dostupné z: < http://rtime.felk.cvut.cz/scheduling-toolbox/manual/>

[Webservices]

LAFON, Y. *Web Services Activity* [online]. c2002, [cit. 2007-04-22]. Dostupné z: <http://www.w3.org/2002/ws/>

[XHTML]

PEMBETON, S., aj. *XHTML 1.0: The Extensible HyperText Markup Language (second edition)* [online]. 26.1.2000, poslední revize 1.8.2002 [cit. 2007-05-15]. Dostupné z: <http://www.w3.org/TR/xhtml1/>

[XML]

QUIN, L. *Extensible Markup Language (XML)* [online]. c1996, poslední revize 8.5.2007 [cit. 2007-05-15]. Dostupné z: <http://www.w3.org/XML/>

Použitý software

[HttpClient]

HttpClient <http://scripts.incutio.com/httpclient/>

[MySQL]

MySQL <http://dev.mysql.com/downloads/>

[PEAR]

PEAR <http://pear.php.net/>

A. Příloha – Popis tabulek databáze

Tabulka user:

Sloupec	Typ	Nulový	Extra	Primární
userID	int	ne	auto inkrement	ano
user	varchar	ne	-	ano
psw	varchar	ne	-	ne
name	varchar	ne	-	ne
email	varchar	ne	-	ne
right	set	ne	-	ne

V tabulce **user** jsou uložena data o zaragistrovaných uživatelích.

Sloupce:

- **userID** – primární klíč tabulky, unikátní identifikátor přiřazený uživateli
- **user** – jméno uživatele používané pro přihlášení do aplikace
- **psw** – uživatelské heslo uložené ve tvaru 32 znakového heše
- **name** – jméno uživatele
- **email** – e-mailová adresa uživatele
- **right** – uživatelova práva (admin, usser, guest)

Tabulka userTemp:Sloupec	Typ	Nulový	Extra	Primární
confirm	varchar	ne	-	ano
user	varchar	ne	-	ano
psw	varchar	ne	-	ne
name	varchar	ne	-	ne
email	varchar	ne	-	ne
right	set	ne	-	ne

Tabulka **userTemp** slouží pro uchovávání dat o uživatelích, kteří se zaregistrovali a ještě si neaktivovali účet nebo o uživatelích, kteří žádají o pomoc se zapomenutým heslem.

Sloupce:

- **confirm** – primární klíč tabulky, obsahuje identifikátor, který byl dočasně uživateli přidělen.
- **user, psw, name, email, right** – řádky shodné s řádky v tabulce user

Tabulka taskset:

Sloupec	Typ	Nulový	Extra	Primární
tasksetID	Int	ne	auto increment	ano
varName	varchar	ne	-	ne
name	varchar	ne	-	ne
description	varchar	ne	-	ne
ownerID	Int	ne	-	ne

V tabulce **taskset** jsou uložena data o sadě úloh.

Sloupce:

- **tasksetID –** primární klíč tabulky, unikátní identifikátor sady, který je jí automaticky přidělen
- **varName –** název proměnné označující sadu úloh v MATLABu
- **name –** jméno sady úloh
- **description –** popis sady úloh
- **ownerID –** identifikátor uživatele, který tuto sadu úloh vytvořil

Tabulka tasksetin:

Sloupec	Typ	Nulový	Extra	Primární
tasksetID	int	ne	-	ano
userID	int	ne	-	ano

Tabulka **tasksetin** reprezentuje vzájemné vazby mezi uživateli a sadami úloh.

Sloupce:

- **tasksetID –** primární klíč**,** identifikátor ukazující na řádek tabulky taskset
- **userID –** primární klíč, identifikátor ukazující na řádek tabulky user

Tabulka task:

Sloupec	Typ	Nulový	Extra	Primární
taskID	int	ne	auto increment	ano
varName	varchar	ne	-	ne
name	varchar	ne	-	ne
procTime	float	ne	-	ne
releaseTime	float	Ano	-	ne
deadline	float	Ano	-	ne
duedate	float	Ano	-	ne
weight	Int	Ano	-	ne
processor	Int	Ano	-	ne
ownerID	Int			ne
		ne	-	

Tabulka **task** uchovává data o úlohách.

Sloupce:

- **taskID –** primární klíč tabulky, unikátní identifikátor úlohy
- **varName –** jméno proměnné, reprezentující úlohu v MATLABu
- **procTime, releaseTime, deadline, duedate, weight, processor –** parametry úlohy vyžadované pro reprezentaci úlohy v TORSCHE.
- **ownerID –** identifikátor sady úloh, pro kterou byla úloha vytvořena

Tabulka taskin:

Sloupec	Typ	Nulový	Extra	Primární
tasksetID	int	ne	-	ano
taskID	int	ne	-	ano

Tabulka taskin reprezentuje vzájemné relace mezi tabulkami taskset a task.

Sloupce:

- **tasksetID –** primární klíč, identifikátor ukazující na sadu úloh v tabulce taskset
- **taskID –** primární klíč, identifikátor ukazující na úlohu v tabulce task

Tabulka prec:

Sloupec	Typ	Nulový	Extra	Primární
tasksetID	int	ne	-	ano
to	int	ne	-	ano
from	Int	ne	-	ano

Tabulka **prec** reprezentuje matici precedenčních omezení používanou v TORSCHE.

Sloupce:

- **tasksetID** – primární klíč tabulky, identifikátor úlohy z tabulky taskset, přiřazuje danou precedenční závislost konkrétní sadě úloh
- **to** – primární klíč tabulky, identifikátor z tabulky task, určuje úlohu, která má být vykonána po dokončení úlohy definované v sloupci **from**
- **from** - primární klíč tabulky, identifikátor z tabulky task, určuje úlohu předcházející úloze definované ve sloupci **to**

Tabulka process:

Sloupec	Typ	Nulový	Extra	Primární
processID	Int	Ne	auto increment	ano
algorithm	varchar	Ne	-	ne
description	varchar	Ne	-	ne
command	varchar	Ne	-	ne
marktime	datetime	Ne	-	ne

V tabulce **process** jaou uložena data popisující zadanou rozvrhovací úlohu.

Sloupce:

- **processID –** primární klíč, unikátní identifikátor zadané rozvrhovací úlohy
- **algorithm –** název zvoleného algoritmu pro rozvrhování
- **description –** uživatelem vytvořený popis rozvrhované úlohy
- **command –** příkaz pro spuštění rozvrhovacího algoritmu v MATLABu
- **marktime –** časový údaj určující dobu, kdy byla úloha zadána k vyřešení

Tabulka processSyntax:

Sloupec	Typ	Nulový	Extra	Primární
processID	int	Ne	-	ano
order	int	Ne	-	ano
type	enum	Ne	-	ne
problem	varchar	Ano	-	ne
processors	int	Ano	-	ne
tasksetID	int	Ano	-	ne
optionsID	int	Ano	-	ne
otherID	int	Ano	-	ne

Tabulka **processSyntax** obsahuje parametry zadané úlohy, které jsou vyžadovány zvoleným rozvrhovacím algoritmem.

Sloupce:

- **processID –** primární klíč tabulky, odkazuje na identifikátor zadané úlohy v tabulce process
- **order –** primární klíč, určuje pořadí vstupního parametru v syntaxi zvoleného algoritmu
- **type –** určuje typ parametru (taskset, problem, processors, options, other) uloženého v daném řádku tabulky
- **problem –** řešený rozvrhovací problém dle standardní notace¨
- **processors –** počet procesorů na kterých je úloha rozvrhována
- **tasksetID –** identifikátor rozvrhované sady úloh z tabulky **taskset**
- **otherID –** odkazuje na identifikátor dat v tabulce **processOther**
- **optionsID –** odkazuje na identifikátor dat v tabulce **processOption**

Tabulka processOther:

Sloupec	Typ	Nulový	Extra	Primární
otherID	int	Ne	auto increment	ano
name	varchar	Ne	-	ne
value	Varchar	Ne	-	ne

V tabulce **processOther** jsou uložena data parametru other v syntaxi zvoleného algoritmu

Sloupce:

- **otherID –** primární klíč, unikátní identifikátor parametru other
- **name –** jméno parametru other
- **value –** hodnota parametru other

Tabulka processOption:

Sloupec	Typ	Nulový	Extra	Primární
optionsID	int	Ne	auto increment	ano
name	varchar	Ne	-	ne
value	Varchar	Ne	-	ne

V tabulce **processOptions** jsou uložena data parametru options v syntaxi zvoleného algoritmu

Sloupce:

- **optionID –** primární klíč, unikátní identifikátor parametru options
- **name –** jméno parametru options
- **value –** hodnota parametru options

Tabulka schedule:

Sloupec	Typ	Nulový	Extra	Primární
scheduleID	int	Ne	auto increment	ano
processID	int	Ne	-	Ano
xmlSchedule	text	Ne	-	Ne
start	datetime	Ne	-	Ne
stop	datetime	Ne	-	Ne
error	datetime	Ano	-	Ne

Tabulka **schedule** obsahuje status řešení dané úlohy

Sloupce:

- **scheduleID –** primární klíč, unikátní identifikátor rozvrhované úlohy, tento sloupec spolu se sloupcem **processID** zároveň slouží pro zámek zabraňující řešení jedné úlohy na více serverech.
- **processID –** primární klíč, identifikátor zadané úlohy z tabulky **process**
- **xmlSchedule –** do tohoto sloupce je v případě úspěšného vyřeršní úlohy uložen rozvrh ve formátu XML
- **start –** údaj určující začátek rozvrhování zadané úlohy
- **stop –** údaj určující konec rozvrhování zadané úlohy
- **error –** v tomto sloupci jsou uloženy případné chyby nastalé při rozvrhování

Tabulka scheduleGantt:

Sloupec	Typ	Nulový	Extra	Primární
ganttID	int	Ne	auto increment	ano
scheduleID	int	Ne	-	ne
gantt	blob	Ne	-	ne
processor	set	Ne	-	ne
width	Int	Ne	-	ne

Tabulka scheduleGantt slouží pro uložení výsledku rozvrhování ve formě Ganttova diagramu.

Sloupce:

- **ganttID –** primární klíč, unikátní identifikátor uloženého Ganttova diagramu
- **scheduleID –** identifikátor rozvržené úlohy odkazující do tabulky **schedule**
- **gantt –** Ganttův diagram ve formě binárních dat
- **processor –** určuje jakým nástrojem byl ganttův diagram vytvořen (MATLAB, PsGantt)
- **width –** šířka Ganttova diagramu

Tabulka scheduleParam:

Sloupec	Typ	Nulový	Extra	Primární
scheduleID	int	Ne	-	ano
Name	varchar	Ne	-	ano
value	text	Ne	-	ne

V tabulce **scheduleParam** jsou uloženy parametry výsledného rozvrhu (hodnota kritérií, doba rozvrhu, aj.).

Sloupce:

- **scheduleID** – primární klíč, identifikátor odkazující na rozvrhovanou úloho do tabulky schedule, pro kterou byly parametry získány
- **name** – jméno parametru
- **value** – hodnota parametru

Tabulka scheduleTask:

Sloupec	Typ	Nulový	Extra	Primární
scheduleID	int	Ne	-	ano
taskID	int	Ne	-	ano
start	float	Ne	-	ne
length	float	Ne	-	Ne
processor	int	Ne	-	Ne

Tabulka **scheduleTask** určuje jak byly jednotlivé úlohy rozvrženy na procesorech.

Sloupce:

- **scheduleID** – primární klíč, identifikátor rozvržené úlohy z tabulky **schedule**
- **taskID** – primární klíč, identifikátor úlohy z tabulky **task**
- **start** – čas přidělení procesoru úloze
- **stop** – čas odebrání procesoru úloze
- **processor** – číslo pocesoru na kterém je úloha vykonávána

Tabulka css:

Sloupec	Typ	Nulový	Extra	Primární
cssID	int	Ne	auto increment	ano
name	varchar	Ne	-	ano
css	text	Ne	-	ne
default	binary	Ne	-	ne
public	binary	Ne	-	ne

Tabulka **css** obsahuje kaskádové styly použité pro vytváření Ganttových diagramů pomocí nástroje PsGantt.

Sloupce:

- **cssID** – unikátní identifikátor zadaného kaskádového stylu
- **name** – jméno kaskádového stylu
- **css** – kaskádový styl uložený ve formátu text
- **default** – určuje zda je tento styl zvolen jako výchozí pro generování Ganttova diagramu
- **public** – určuje zda je tento styl dostupný i ostatním uživatelům

Tabulka cssin:

Sloupec	Typ	Nulový	Extra	Primární
userID	int	Ne	-	ano
cssID	int	Ne	-	ano

Tabulka **cssin** reprezentuej relace mezi tabulkou **user** a tabulkou **css**.

Sloupce:

- userID – primární klíč, identifikátor uživatele z tabulky **user** jenž vlastní daný styl
- cssID – primární klíč, identifikátor kaskádového stylu z tabulky **css**

Tabulka css2gantt:

Sloupec	Typ	Nulový	Extra	Primární
ganttID	int	Ne	-	ano
cssID	int	Ne	-	ano

Tabulka **css2gantt** reprezentuje relace mezi tabulkami **css** a **scheduleGantt**.

Sloupce:

- **ganttID** – primární klíč, identifikátor Ganttova diagramu z tabulky **scheduleGantt** vytvořeného za použití daného kaskádového stylu
- **cssID** – primární klíč, identifikátor kaskádového stylu z tabulky **css**

Tabulka machine:

Sloupec	Typ	Nulový	Extra	Primární
machineID	int	Ne	-	ano
url	varchar	Ne	-	ano
path	Varchar	Ne	-	ne
status	enum	Ne	-	ne

Tabulka **machne** obsahuje seznam Matlab sereverů použitých pro rozvrhování úloh.

Sloupce:

- **machineID –** primární klíč, unikátní identifikátor Matlab serveru
- **url –** internetová adresa Matlab serveru
- **path –** cesta na Matlab serveru k spuštěnému skriptu
- **status –** status Matlab serveru (active, off)

Tabulka solver:

Sloupec	Typ	Nulový	Extra	Primární
processID	int	Ne	-	ano
machineID	int	Ne	-	ne

Tabulka **solver** slouží pro přidělování úloh z tabulky **process** na Matlab servery z tabulky **machine**.

Sloupce:

- **processID –** primární klíč, identifikátor úlohy z tabulky **process** přiřazené danému Matlab serveru
- **machineID –** identifikátor Matlab serveru z tabulky **machine** na kterém je úloha rozvrhována.

Tabulka threadLock:

Sloupec	Typ	Nulový	Extra	Primární
machineID	int	Ne	-	ano
threadID	int	Ne	-	ne

Tabulka threadLock slouží pro zámek vláken a procesů spouštěných na Matlab serverech.

Sloupce:

- **machineID** – primární klíč, identifikátor Matlab serveru z tabulky **machine**
- **threadLock –** identifikátor vlákna nebo procesu na Matlab serveru

Tabulka log:

Sloupec	Typ	Nulový	Extra	Primární
time	Datetime	Ne	-	ano
log	varchar	Ne	-	ano

Tabulka **log** slouží pro ukládání informací o průběhu řešení úloh.

Sloupce:

- **time** – primární klíč, čas zaznamenáné události
- **log** – popis zaznamenané události

B. Obsah CD:

Přiložené CD obsahuje tyto adresáře a soubory

- **dp_jan_martinsky.doc** – tento dokument ve verzi .doc
- **dp_jan_martinsky.pdf** – tento dokument ve verzi.pdf
- **databaze.txt** – textový soubor s SQL příkazem pro vytvoření databáze
- **/SOAPserver –** skripty SOAP serveru
- **/Matlabserver –** skripty Matlab serveru
- **/Web –** skripty webového rozhranní