Problematika plánování úloh v prostředí Gridu

Dalibor Klusáček, Luděk Matyska, Hana Rudová
Fakulta informatiky, Masarykova univerzita
{xklusac, ludek, hanka}@fi.muni.cz

Abstrakt: Efektivní plánování v prostředí Gridu představuje komplexní problém, který není v současné době uspokojivě vyřešen. Nalezení optimálního rozvrhu, tj. přiřazení úloh v čase na dostupné zdroje představuje NP úplný problém, který je pro větší množství úloh v rozumném čase neřešitelný. Vhodným postupem je proto hledání suboptimálních řešení, kde existují rychlejší algoritmy. Produkční plánovací systémy pak představují opačný extrém, neboť zpravidla používají pouze jednoduché algoritmy založené na plánování pomocí (prioritních) front. Nasazením pokročilých plánovacích technik, jako jsou např. heuristiky pro optimalizaci globálního rozvrhu, je možné tento rozvrh významně zkvalitnit, zkrátit čas dokončení úloh a optimalizovat využití Gridu.

V tomto příspěvku se věnujeme aplikací známých plánovacích heuristik založených na lokálním prohledávání a řídících pravidlech v dynamickém prostředí Gridu. Tyto heuristiky byly původně navržené pro statické prostředí, prezentujeme proto jejich modifikace, které je možné efektivně nasadit i pro tvorbu rozvrhů při dynamicky se měnícím počtu úloh, typickým pro Gridové prostředí. Pro testování vlastností těchto plánovacích algoritmů jsme adaptovali a dále rozšiřujeme flexibilní simulační prostředí GridSim. To umožňuje modelovat typické vlastnosti úloh i gridového prostředí a studovat scénáře nasazení gridových plánovačů. V příspěvku prezentujeme simulaci plánování dynamicky přibývajících paralelních úloh na heterogenních výpočetních zdrojích, globální rozvrh je vytvářen řídícími pravidly a optimalizován pomocí Tabu prohledávání. Tímto způsobem jsme dosáhli výrazného zlepšení celkového rozvrhu a zmenšení celkového zpoždění úloh.

1. Úvod

Grid [FK04] bývá chápán jako distribuovaný a heterogenní počítačový systém poskytující uživatelům různorodé služby – výpočetní a úložné kapacity, specifické zpracování dat, distribuovaný sběr a zpracování informací atd. – prostřednictvím různých zdrojů. Cílem je maximalizovat využití zdrojů a zároveň poskytovat netriviální kvalitu služby pro koncové uživatele Gridu. Plánovací systémy se snaží tyto požadavky uspokojit vhodným rozvržením úloh na jednotlivé zdroje. Problém plánování je však velice složitý, neboť prostředí Gridu je heterogenní a především dynamické. Hledat v takovémto případě optimální řešení je nemožné, neboť se jedná o NP úplný problém, který není v rozumném čase řešitelný.

Současně používané dávkové systémy jako Condor, LSF nebo PBS[1] proto tuto problematiku často řeší značně triviálními postupy, které jsou založeny zejména na principu (prioritních) front úloh. I pokročilejší experimentální nástroje jako Grid Service Broker, GridWay nebo GRMS[2] používají podobné postupy. Ty však často negenerují dobré výsledky. V této práci navrhujeme nový přístup, vycházející z principu rozvrhu. Pro jeho tvorbu a optimalizaci použijeme pokročilé plánovací techniky [GK03]. Námi navržené algoritmy jsou založeny na řídících pravidech [Pin02] a lokálním prohledávání [Pin02, Klu06]. Tyto heuristiky byly původně určeny pro statické problémy, kdy je celý systém neměnný a počet plánovaných úloh je dopředu znám. Pro nasazení v prostředí Gridu jsme tyto heuristiky upravili tak, aby fungovaly i v dynamicky se měnícím prostředí.

Současně s novými algoritmy byl vyvinut simulátor Gridu [Klu06], který byl využit na

[1] Condor viz http://www.cs.wisc.edu/condor/, LSF viz http://www.platform.com/, PBS viz http://www.altair.com/

[2] Grid Service Broker viz http://gridbus.org/broker/, GridWay viz http://asds.dacya.ucm.es/GridWay/, GRMS viz http://www.gridlab.org/grms

modelování a ověření funkčnosti těchto plánovacích algoritmů. Simulátor umožňuje testovat navržené algoritmy při různých parametrech úloh, různém počtu dostupných zdrojů a různých optimalizačních kritériích, v situaci, kdy úlohy v systému dynamicky přibývají a zároveň již dokončené úlohy v systému dále nefigurují. Funkčnost simulátoru i navržených algoritmů byla experimentálně ověřena a vybrané výsledky jsou zde prezentovány.

Následující kapitola obsahuje popis problému, poté následuje popis simulátoru. V další kapitole jsou popsány navržené algoritmy a dále jsou prezentovány vybrané experimentální výsledky. V závěru je shrnuta dosavadní práce a navrženy budoucí cíle výzkumu.

2. Popis problému

Problém plánování úloh na Gridu [FMR05] lze charakterizovat množinou zdrojů, množinou úloh, parametry prostředí a optimalizačním kritériem. Množinou zdrojů zde budeme rozumět množinu dostupných strojů. Tyto stroje mohou být charakterizovány mnoha parametry, například počtem a rychlostí procesorů, velikostí operační paměti, dostupným softwarem atd.

Úloha je obvykle primárně charakterizována počtem procesorů, které vyžaduje pro svůj běh. Pokud je tento počet větší než jedna, mluvíme zde o paralelních úlohách. Důležitým parametrem je znalost případně neznalost očekávané doby trvání úlohy pro konkrétní stroj. Častým parametrem úlohy bývá termín dostupnosti nebo termín dokončení, tedy čas, kdy nejdříve může být úloha zpracována a čas kdy má být dokončena. Kromě toho může úloha požadovat konkrétní software, minimální volnou kapacitu disku nebo operační paměti, případně dostupnost vstupních dat, minimální kapacitu síťového propojení atd.

Stroj bývá charakterizován počtem a typem procesorů, jejich rychlostí, velikostí operační paměti nebo diskovou kapacitou, dostupným programovým vybavením případně operačním systémem, instrukční sadou atd.

Existuje mnoho způsobů, jak jednotlivým úlohám přidělit vhodné zdroje. Pro jejich porovnání potřebujeme metriku, kterou zpravidla vyjádříme pomocí tzv. objektivní funkce. Hodnota této objektivní funkce pak představuje optimalizační kritérium řešeného problému. Mezi nejčastěji používaná kritéria patří minimalizace času dokončení poslední úlohy, minimalizace celkového nebo maximálního zpoždění úloh nebo maximalizace využití strojů. Tato kritéria lze navzájem kombinovat, např. prostřednictvím přiřazením váhy jednotlivým kritériím.

Parametry prostředí určuje množství faktorů jakými mohou být parametry počítačové sítě, spolehlivost atd. Velký význam hraje to, zda se jedná o prostředí statické či dynamické. Statickým prostředím rozumíme takovou situaci, kdy jsou všechny úlohy, stroje a ostatní činitele známy před začátkem plánovacího procesu a v jeho průběhu se nemění. Oproti tomu dynamické prostředí je charakterizováno přibýváním úloh a změnami v počtu a dostupnosti strojů v průběhu času, s čímž se musí plánovací proces vyrovnat. Je zřejmé, že Grid je typickým dynamickým prostředím. Další rozšíření představuje zavedení přerušitelnosti úlohy nebo migrace běžících úloh mezi stroji.

3. Plánování úloh v prostředí Gridu

Plánováním úloh [Pin02] v prostředí Gridu obecně rozumíme alokaci úloh na dostupné zdroje (zde stroje) v čase tak, aby se splnila zadaná kritéria, například minimalizovala doba dokončení poslední úlohy nebo maximalizovalo využití zdrojů. Důraz je kladen na uspořádání úloh v čase.

Výstupem plánovacího procesu je v jednodušším případě fronta úloh – jedna nebo více, případně pro každý zdroj samostatná – nebo rozvrh. Rozvrh je oproti frontě komplexnější datová struktura obsahující nejen informace o umístění úloh na zdroje, ale i o času jejich zpracování.

Nalezení optimálního rozvrhu vzhledem k nějakému kritériu je NP úplný problém [BK06], což v praxi znamená, že již pro velice malý počet úloh a zdrojů je nalezení takovéhoto rozvrhu nerealizovatelné v rozumném čase [BK05]. Z tohoto důvodu se v praxi nehledá optimální řešení, nýbrž řešení suboptimální, které však jsme schopni naleznout v akceptovatelném čase. K tomu se používá řada heuristik, mezi které patří řídící pravidla (dispatching rules) i lokální prohledávání (local search).

V drtivé většině současných operačních i experimentálních plánovacích systémů se využívá plánování pomocí front do nichž jsou umisťovány úlohy před zpracováním na stroji. Těchto front může být více – například každá s jinou prioritou – pak hovoříme o prioritních frontách. Díky tomu, že rozvrh poskytuje informace o času zpracování každé úlohy, mohou plánovací algoritmy tuto informaci využít pro změny a optimalizaci stávajícího rozvrhu, rezervaci zdrojů (advanced reservation), práci s interaktivními úlohami, atd. Podmínkou je, aby byly k dispozici relativně přesné informace o předpokládané době běhu úlohy na daném stroji.

4. Simulátor

Pro návrh a implementaci plánovacích algoritmů bylo zvoleno simulační prostředí GridSim [BM02], které je napsáno v jazyce Java. Existuje celá řada obdobných nástrojů jako Bricks, SimGrid, Simbatch nebo Microgrid, ale GridSim se vyznačuje dobrou dokumentací a snadnou rozšiřitelností. GridSim umožňuje simulovat prostředí Gridu s jeho zdroji, úlohami a dalšími entitami. Tyto entity spolu navzájem komunikují prostřednictvím zasílání správ, což do značné míry odpovídá realitě.

V tomto simulačním prostředí pak byl vyvinut simulátor, který podstatně rozšířil standardní funkcionalitu GridSimu o nové vlastnosti a nové entity. Z důležitých vlastností jmenujme dynamické přibývání úloh v čase nebo podporu zpracování paralelních úloh. Nové entity pak představují například centralizovaný plánovač nebo systém přijímání úloh (job submission system) zodpovědný za správu úloh a komunikaci s plánovačem.

Simulátor byl navržen jako modulární, aby umožňoval snadnou rozšiřitelnost pro další výzkum. Všechny důležité entity jako úloha, zdroj, plánovací algoritmus jsou modelovány samostatnou třídou. Entity jež spolu navzájem komunikují, jako například plánovač a systém přijímání úloh, pak v sobě oddělují komunikační funkcionalitu od vnitřních metod. To vše dohromady umožňuje případné změny, jako například změnu plánovacího algoritmu nebo typu úloh, realizovat lokálně s minimálním zásahem do celého simulátoru.

5. Algoritmy

V našem výzkumu jsme se zaměřili na vývoj plánovacích algoritmů založených na lokálním prohledávání a řídících pravidlech. Algoritmy založené na myšlence řídících pravidel představují jednoduché heuristiky, které plánují úlohy v tom pořadí v jakém úlohy přicházejí do systému. V závislosti na použitém řídícím pravidle je pro tuto novou úlohu vybrán stroj a úloha je zařazena mezi ostatní úlohy čekající na zpracování na tomto stroji. Způsob výběru stroje a pořadí úlohy mezi ostatními úlohami je dáno typem řídícího pravidla, které odpovídá sledovanému optimalizačnímu kritériu. Optimalizační kritérium v tomto případě ukládalo

minimalizovat *celkové nezáporné zpoždění všech úloh*. Pro toto kritérium jsme navrhli dvě řídící pravidla – *Minimum Tardiness Earliest Due Date (MTEDD)* a *Minimum Tardiness Earliest Release Date (MTERD)*. MTEDD pro novou úlohu vybírá stroj s aktuálně nejmenší hodnotou předpokládaného celkového zpoždění úloh. Úloha je zařazena mezi čekající úlohy v závislosti na svém termínu dokončení (due date), a to tak, že úloha s dřívějším termínem dokončení bude zpracována dříve. MTERD pro novou úlohu vybírá stroj s aktuálně nejmenší hodnotou předpokládaného celkového zpoždění úloh. Úloha je zařazena mezi čekající úlohy v závislosti na svém termínu dostupnosti (release date), a to tak, že úloha s dřívějším termínem dostupnosti bude zpracována dříve.

Lokální prohledávání na rozdíl od řídících pravidel pracuje s existujícím rozvrhem. Nad tímto rozvrhem jsou pak prováděny lokální změny (např. výměna pořadí dvou úloh naplánovaných na stejném stroji) a je zkoumán jejich vliv na kvalitu řešení. V případě, že lokální změna rozvrhu vede k lepšímu řešení, je tato změna přijata, v opačném případě je většinou zamítnuta. Dočasný přechod k horšímu řešení umožní prozkoumat větší prostor řešení a nalézt tak finálně řešení mnohem lepší. Z existujících variant lokálního prohledávání byl implementován Horolezecký algoritmus [Pin02, Klu06], Simulované žíhání [HJJ03, Klu06] a Tabu prohledávání [Gen03, GP05, Klu06]. Nejlepší experimentální výsledky byly dosaženy v případě použití *Tabu prohledávání (Tabu Search – TS)*. Tabu prohledávání zavádí tzv. Tabu seznam, což je seznam několika posledních změn v rozvrhu, které nesmí být v následujících krocích algoritmu opakovány. Takto je možné zbránit zacyklení plánovacího algoritmu. Pro tvorbu iniciálního rozvrhu byla použita výše zmíněná řídící pravidla MTEDD a MTERD. *Lokální změnu* v existujícím rozvrhu představoval výběr zpožděné úlohy ze stroje s předpokládaným největším celkovým zpožděním úloh a její přesun na stroj s menším očekávaným zpožděním.

6. Experimentální výsledky

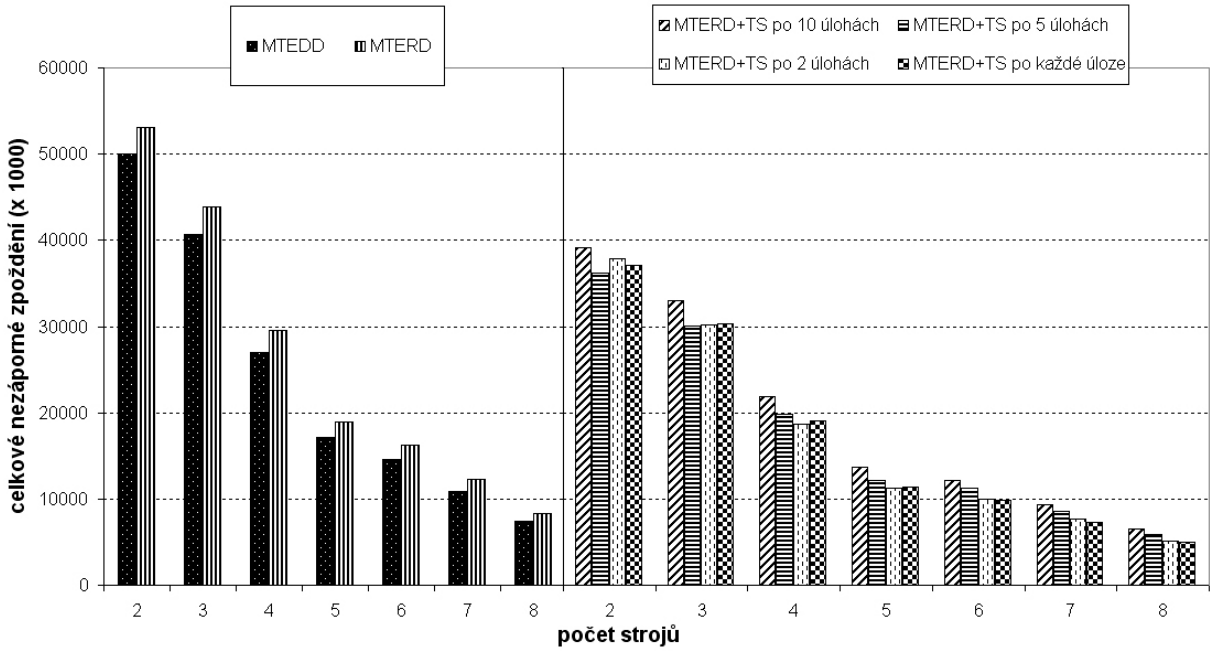
Experimentální ověření algoritmů jsme provedli pro gridový systém s následujícími vlastnostmi. Systém průběžně přijímá úlohy a ty jsou s využitím řídících pravidel umisťovány do rozvrhu. Ve vhodný okamžik je spuštěno Tabu prohledávání (TS), které optimalizuje aktuální rozvrh. Plánované úlohy jsou jedno i víceprocesorové (paralelní), počítače mohou v jediném okamžiku zpracovávat více úloh, a to až do své kapacity (zpravidla počet procesorů či jader). Celkově bylo naplánováno 1000 úloh. Počet dostupných strojů se pro jednotlivé experimenty lišil. Výsledky jednoho experimentu byly získány jako průměrná hodnota z výsledků dvaceti různých datových sad. Tyto datové sady byly generovány synteticky s parametry uvedenými v *Tabulce 1*. Každý stroj má 4 CPU, různé stroje se liší rychlostí CPU, která ovlivňuje dobu zpracování úloh. Pokud po dokončení nějaké úlohy hrozí snížení vytížení daného stroje, je podle připraveného rozvrhu vybrána úloha a je odeslána na tento stroj, kde poté bude zpracována. Úlohy již odeslané na stroj se nepřesunují na jiný stroj (není podporována migrace). Cílem plánovače je snížit celkové nezáporné zpoždění všech úloh. Optimalizace pomocí TS je prováděna postupně po každé úloze, dvou, pěti nebo deseti nových úlohách v systému. Experimenty byly prováděny na počítači s procesorem Intel Pentium 4 2.6 GHz s 512 MB RAM.

parametry úlohy	*min*	*max*	*průměr*	*směrodatná odchylka*
čas příchodu do systému	5	14958	7476	4168
termín dostupnosti	27	16631	7993	4181
termín dokončení	517	19438	9981	4265
počet požadovaných CPU	1	4	2,19	1,16
výpočetní délka	1	14996	4506	3554

Tabulka 1. parametry použité datové sady.

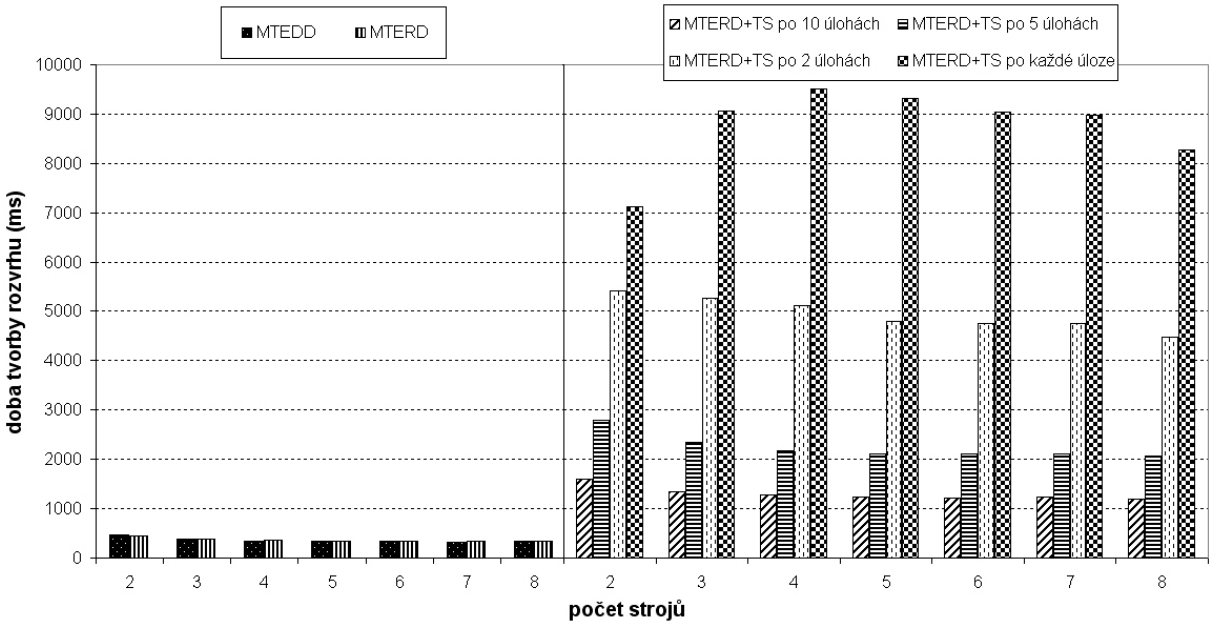
6.1. Diskuze

Z Obrázku 1 vyplývá, že se vzrůstajícím počtem dostupných strojů přirozeně klesá celkové zpoždění. TS také ve všech případech dokázalo optimalizovat rozvrh tak, aby celkové nezáporné zpoždění úloh bylo menší, než v případě použití pouze MTERD či MTEDD. *Obrázek 2* oproti tomu ukazuje, že obě řídící pravidla jsou oproti TS výrazně rychlejší. Je to dáno nutností opakovaně porovnávat hodnoty celkového nezáporného zpoždění dvou rozdílných rozvrhů při použití TS. Tato porovnání jsou prováděna mnohokrát, proto je doba tvorby rozvrhu větší oproti MTERD a MTEDD.



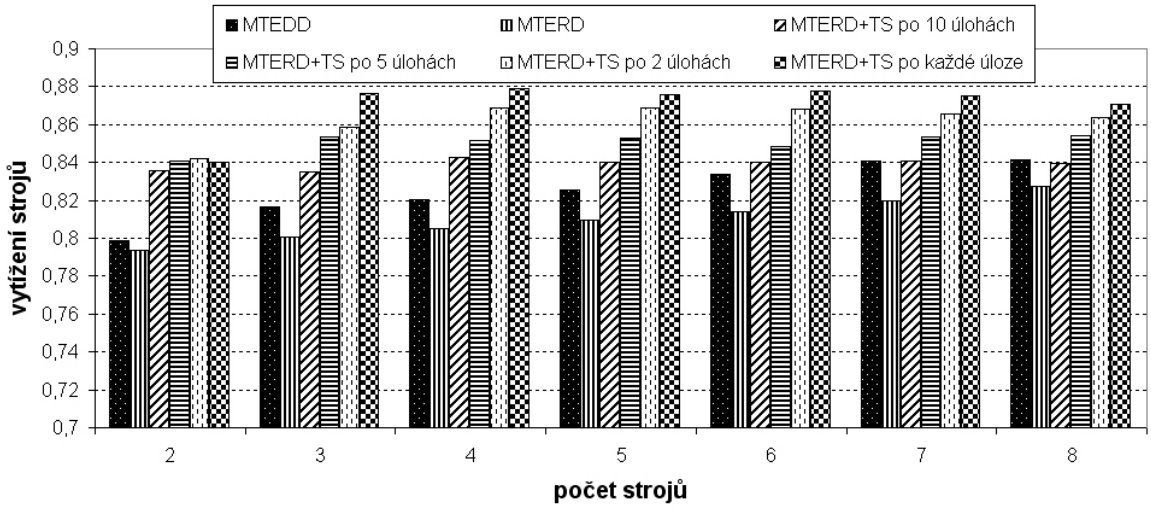
Obrázek 1. Celkové nezáporné zpoždění pro 1000 úbh při různém počtu strojů.

Frekvence vyvolání optimalizační procedury TS ovlivňuje celkovou dobu nutnou pro vypočtení rozvrhu. Je zřejmé, že optimalizace po každé nové úloze je časově náročnější, než při optimalizaci po každé desáté úloze.



Obrázek 2. Doba tvorby rozvrhu pro 1000 úloh při různém počtu strojů.

Kromě zlepšení objektivní funkce došlo při použití TS optimalizace ke zvýšení vytížení zdrojů, jak ukazuje *Obrázek 3*. Pro výraznější zvýšení vytížení by pak bylo potřeba modifikovat navrženou proceduru TS tak, aby kupříkladu umožnila spouštět nenáročné úlohy (počet potřebných CPU a výpočetní délka) na volných procesorech v době, kdy jiná náročná úloha čeká na uvolnění potřebného počtu procesorů. V takovéto situaci totiž nyní volné procesory využity nejsou.



Obrázek 3. Vytížení strojů

7. Závěr

Cílem naší práce bylo vyvinout pokročilé plánovací algoritmy schopné řešit dynamické problémy plánování úloh na Gridu. Navrhli jsme jednoduchá ale dostatečně účinná řídící pravidla a optimalizační algoritmus Tabu prohledávání, který byl schopen zlepšovat kvalitu dosaženého řešení. Vedle vlastního návrhu algoritmu budujeme i komplexní simulační prostředí, které umožňuje snadno modelovat nejrůznější problémy a situace při plánování úloh na Gridech. V tomto simulátoru byly námi navržené algoritmy experimentálně vyzkoušeny.

V současné době pracujeme na implementaci algoritmů založených na principech

metody Backfilling [FRS05], která je typickým představitelem pokročilejších technik pro plánování pomocí front. To nám umožní srovnat kvalitu dosažených výsledků při použití algoritmu backfilling a při použití rozvrhu a jeho optimalizace pomocí lokálního prohledávání. V budoucnu plánujeme rozšířit simulátor o simulaci sítě a simulaci různých výpadků. Zatímco současné plánovací algoritmy počítají se znalostí doby trvání úloh, našim plánem je porovnat chování jednotlivých algoritmů v situaci, kdy neznáme dobu trvání úloh, resp. kdy se výrazně liší plánovaná a skutečná doba výpočtu (např. dojde k selhání výpočtu, odhad je nepřesný atd.). Pracujeme také na rozšíření modelu Gridu ze současného malého počtu strojů na stovky strojů. První testy ukazují, že simulátor bez problémů zvládá simulaci systému se 150 stroji a další rozšíření by mělo být poměrně jednoduché. Do budoucna chceme využit existující záznamy z běhu reálných úloh (workloads) vzhledem k tomu, že současné testy pracují pouze s uměle generovanými problémy.

Poděkování

Tato práce je podporována výzkumným záměrem MSMT0021622419 Ministerstva školství, mládeže a tělovýchovy ČR a projektem 201/07/0205 Grantové agentury ČR.

Literatura

[BK05] Edmund K. Burke, Graham Kendall. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer 2005.

[BK06] Peter Brucker, Sigrid Knust. *Complexity results for scheduling problems*. Universitaet Osnabrueck, URL http://www.mathematik.uni-osnabrueck.de/research/OR/class/

[BM02] Rajkumar Buyya and Manzur Murshed. *GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing*. The Journal of Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press, 2002.

[FK04] I. Foster, C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Second edition, Morgan-Kaufman, 2004.

[FMR05] Pavel Fibich, Luděk Matyska, Hana Rudová. *Model of Grid Scheduling Problem*. Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing. Papers from the AAAI-05 workshop. Technical report WS-05-03, AAAI Press, 2005.

[FRS05] D. Feitelson, L. Rudolph, and U. Schwiegelshohn. Parallel job scheduling – a status report. Job Scheduling Strategies for Parallel Processing 2004, LNCS 3277, pages 1-16, 2005.

[Gen03] Michel Gendrau. *An introduction to tabu search*. V [GK03] pages 37-54, chapter 2.

[GK03] Fred Glover, Gary A. Kochenberger. *Handbook Of Metatheuristics*. Kluwer Academic Publishers, 2003.

[GP05] Michael Gendreau Jean-Yves Potvin. *Tabu search*. V [BK05] pages 165-186, chapter 6.

[HJJ03] Darrall Henderson, Sheldon H. Jacobson, Alan W. Johnson. *The theory and practise of simulated annealing*. V [GK03] pages 287-319, chapter 10.

[Klu06] Dalibor Klusáček. *Plánování úloh v paralelním a distribuovaném prostředí*, diplomová práce, Fakulta informatiky, Masarykova univerzita, 2006.

[Koc02] Waldemar Kocjan. *Dynamic scheduling state of the art report*. SICS Technical Report T2002:28, 2002.

[Pin02] Michael Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall, second edition, 2002.

[Pin05] Michael Pinedo. *Planning and Scheduling in Manufacturing and Services*. Springer, 2005.