Porovnávání rychlosti a přesnosti řešení některých úloh lineární algebry na počítačích Silicon Graphics Crimson a Cray YMP EL

Tůma, Miroslav
1993

Porovnání rychlosti a přesnosti řešení některých úloh lineární algebry na počítačích Cray Y-MP EL a Silicon Graphics Crimson.

Miroslav Tůma, Miroslav Rozložník *

21. října 1994

Tento článek přináší porovnání výkonnosti počítače Cray Y-MP EL instalovaného ve FzÚ AV ČR a pracovní stanice Silicon Graphics Crimson s procesorem R4000 z ÚIVT AV ČR pro některé základní úlohy lineární algebry. Je zde zahrnuto řešení rozsáhlejších lineárních systémů pro případ symetrické a pozitivně definitní matice, symetrické indefinitní matice a nesymetrické matice soustavy. Dále je provedeno porovnání pro symetrický problém vlastních čísel a SVD rozklad. Porovnávané parametry zahrnují čas spotřebovaný procesorem pro výpočet, přesnost řešení a velikost rezidua systému. Matice testovaných lineárních soustav vznikají v závažných fyzikálních, chemických a ekonomických aplikacích, které jsou stručně uvedeny, a jsou vzaty ze souboru Harwell-Boeing (viz [8]).

1 Úvod.

Tento článek přináší porovnání výkonnosti dvou počítačů používaných v AV ČR pro vědecko-technické výpočty. Testy zahrnují řešení vybraných základních úloh lineární algebry s maticemi, které interpretujeme jako řídké či husté a používáme odpovídající algoritmy. Protože tyto problémy tvoří součást velkého množství vědecko-technických výpočtů, snaží se toto porovnání přispět k lepšímu poznání reálné výkonnosti uvedené výpočetní techniky. Nejprve shrňme některé základní známé technické a teoretické údaje o použité výpočetní technice.

Nechť Ψ značí strojovou přesnost definovanou jako $1^+ - 1$, kde 1^+ je následník čísla 1 v použité aritmetice. Položme $\Psi = \beta^{-p+1}$. Námi použitá pracovní stanice SGI Crimson s procesorem R4000 používá aritmetiku podle standardu IEEE, kde $\beta = 2$ a $p = 53$. Použitý počítač Cray Y-MP EL má $\beta = 2$ a $p = 50$. Definujme dále zaokrouhlovací jednotku $\mathbf{u}$ jako největší číslo, které přičteno k číslu 1 je nezmění. Kromě strojové přesnosti tato jednotka závisí také na použitém způsobu zaokrouhlování v počítačové aritmetice. Pro způsob zaokrouhlování k nejbližšímu číslu u pracovních stanic platí $\mathbf{u} = \frac{1}{2}\Psi$. Pro použitý typ počítače Cray platí $\mathbf{u} = \Psi$. Tyto strojové konstanty a takt procesoru jsou uvedeny v Tabulce 1.

Hodnoty	Cray Y-MP EL	SGI Crimson
Ψ	1.6D-15	2.2D-16
$\mathbf{u}$	1.6D-16	1.1D-16
Takt	33.3 MHz	50 MHz

Tabulka 1: Porovnání strojové přesnosti, zaokrouhlovací jednotky, a taktu procesoru pro použité počítače Cray Y-MP EL a SGI - Crimson.

Základní standardní porovnání počítačů pro řešení generovaných hustých lineárních systémů je publikováno v pravidelně aktualizované zprávě [5]. V Tabulce 2 jsou uvedeny údaje o výkonu námi použitých počítačů na testovacích úlohách o dimenzi 100 a 1000 označené zde jako L100 a L1000. Dále

*Skupina aplikované lineární algebry, ÚIVT AV ČR, Pod vodárenskou věží 2, 182 07 Praha 8 - Libeň, Česká republika.
Vypracováno v rámci grantu GA AV ČR č. 23003 a grantu GA ČR č. 201/93/0067.

je zde uvedena teoretická špičková rychlost odpovídající výsledku na ideálně vektorizovatelné úloze pro Cray resp. s ideálně vytíženým procesorem pro SGI Crimson. Detaily o použitých kompilátorech a podmínkách testu mohou být nalezeny v [5]. Výsledky jsou uvedeny v miliónech operací v pohyblivé řádové čárce za sekundu (MFLOPS).

Výkon	Cray Y-MP EL	SGI Crimson
L100	32 MFLOPs	16 MFLOPs
L1000	107 MFLOPs	32 MFLOPs
Špičkový výkon	133 MFLOPs	50 MFLOPs

Tabulka 2: Porovnání tabulkových výkonů počítačů.

Důvodem použití dalších testovacích úloh je především význam řešení rozsáhlých a řídkých lineárních systémů v technické praxi, vznikajících často linearizací diskretizovaných problémů. Kvalitní testovací problémy založené na řešení hustých lineárních systémů jsou běžně k dispozici, ale o aplikačních úlohách mohou vypovídat někdy málo. Náš problém je k realitě často blíže, jak o tom ostatně svědčí project Perfect Benchmark Club (viz [2]), nicméně není a nemůže být také vyčerpávající.

Pravděpodobně největší výhodou počítače Cray Y-MP EL je vektorový procesor, který umožňuje efektivní provádění vektorových operací. To se projeví především při práci s hustými maticemi, kde se obvykle provádějí operace s hustými řádkovými či sloupcovými vektory. Pracovní stanice SGI Crimson vektorový procesor nemá, nicméně její skalární jednotka R4000 typu RISC je velmi účinná.

Všechny použité programy byly napsány v normě Fortranu 77 a počítány ve dvojité přesnosti. Některé řešiče byly napsány autory, ostatní programy pocházejí ze souborů Linpack (viz [6]) a Lapack (viz [1]). Bližší údaje jsou uvedeny v následující kapitole při rozboru jednotlivých případů. Hlavní programy načítající data a provádějící výpočet rezíduí a přesnosti řešení byly napsány autory.

Pro překlad na obou počítačích jsme použili maximální stupeň optimalizace programu zahrnující výpočet konstant v průběhu kompilace, odstranění nadbytečného počítání výrazů a nadbytečných proměnných, optimalizace vytížení registrů a alokace paměti, pořadí operací, pohybů invariantních částí kódu vzhledem k cyklům aj. V případě vektorového počítače Cray Y-MP EL se navíc uplatnilo plánování instrukcí pro vektorové funkční jednotky. Detaily jsou uvedeny v příručkách k oběma počítačům. Pro referenci uvedeme nyní parametry překladu zdrojového programu **program.f** použité na jednotlivých počítačích. Na pracovní stanici SGI Crimson jsme použili syntaxi:

```
f77 -O3 program.f.
```

Na počítači Cray Y-MP EL jsme použili příkaz:

```
cf77 -Zv -Wd"-e6" -Wf"-emx -dp -a static -o aggress" program.f.
```

Časové údaje na pracovní stanici SGI Crimson byly měřeny pomocí standardní procedury **ETIME**. Na počítači Cray jsme použili proceduru **SECOND**. Jsou udány v sekundách. Nezměřené časy pro některé matice odpovídají času většímu než 200s. Označíme-li měřené jádro programu po načtení matice jako volání **CALL PROGRAMCORE**, pak naše měření času pro SGI Crimson můžeme znázornit následujícím způsobem:

```
BEFORE = ETIME(T1,T2)
BEFORE = ETIME(T1,T2)
CALL PROGRAMCORE
AFTER = ETIME(T3,T4)
WRITE(*,*)' CRIMSON USERTIME =', T3-T1
```

Pro měření času na počítači Cray bylo použito následující schéma:

```
TO = SECOND()
OVERHEAD = SECOND() - TO
BEFORE = SECOND()
CALL PROGRAMCORE
```

```
AFTER = SECOND()
WRITE(*,*)' CRAY USERTIME =', AFTER-BEFORE-OVERHEAD
```

Matice lineárních systémů jsou vzaty z kolekce Harwell-Boeing (viz [8]). Aplikační oblasti, ve kterých matice vznikly, nyní stručně uvedeme pro představu o šíři záběru tohoto porovnání. Matice BP800 vznikla v simplexové metodě lineárního programování pro ekonomické plánování. Soustava s maticí FS5413 vznikla při řešení úlohy znečištění ovzduší zahrnující chemickou kinetiku a dvoudimenzionální rovnici transportu. Soustava s maticí GRE1107 vznikla při simulaci chování počítačových systémů. JPWH991 vznikla při simulaci chování elektrického obvodu. Soustavy s maticemi BC, BRNO a NOS pocházejí ze strukturálního inženýrství. Matice BUS vznikly při návrhu rozvodných sítí. GR3030 má zdroj v devítibodové diskretizaci na jednotkovém čtverci s Dirichletovými okrajovými podmínkami. Matice WEST pocházejí z úloh chemického modelování. MAHINDAS je produktem ekonomického modelování. MCFE pochází z astrofyzikální aplikace. NNC666 vznikla při počítání chlazení jaderného reaktoru. PORES2 pochází z aplikace modelování nádrže na kapaliny. Matice ORSIRR a SHERMAN vznikly při modelování ropných úložišť. Pravé strany všech systémů byly spočítány pomocí známého řešení se všemi komponentami jednotkovými.

2 Řešení soustav lineárních rovnic.

2.1 Symetrický pozitivně definitní případ.

Uvažujme problém řešení soustavy lineárních algebraických rovnic

$$Ax = b \tag{2.1}$$

kde $A \in R^{N,N}$ je symetrická pozitivně definitní (SPD) matice a $b \in R^N$ je vektor pravé strany. Standardním postupem při řešení SPD soustav je výpočet Choleského rozkladu $A = R^T R$, kde R je horní trojuhelníková matice a pomocí vypočteného faktoru provedený zpětný chod (viz klasické publikace [11], [17]). Mezi nejefektivnější implementace řešení soustav Choleského rozkladem patří jeho implementace v lineárních knihovnách Linpack (viz [6]) a Lapack (viz [1]). Pro porovnání jsme použili programy DPOFA a DPOSL implementované v souboru Linpack (viz [6]). Matice soustav jsou uložené v standardním dvourozměrném poli o rozměrech N x N. Výsledky shrnuté v Tabulce 3 dokumentují, že v tomto hustém případě je Choleského dekompozice snadno vektorizovatelná a CPU-časy naměřené na počítači Cray jsou výrazně lepší než časy na sekvenční pracovní stanici.

	Rozměr	P. nenul.	Čas CPU		Norma chyby		Norma rezidua	
MATICE	N	prvků NZ	Cray	Crimson	Cray	Crimson	Cray	Crimson
1138BUS	1138	2596	27.88	66.29	1.05–06	7.60–10	2.85–08	4.61–11
494BUS	494	1080	4.44	4.75	1.25–08	2.16–11	1.42–09	1.29–11
662BUS	662	1568	8.36	11.7	8.51–09	3.89–12	1.62–10	9.75–13
685BUS	685	1967	9.00	13.07	1.50–09	3.85–13	3.11–10	8.29–12
BRNO974	974	12340	19.61	41.03	1.14–09	1.52–12	2.93–08	1.49–10
GR3030	900	4322	16.45	31.61	6.35–11	2.94–14	5.44–12	4.92–14

Tabulka 3: Výsledky porovnání pro hustou implementaci Choleského rozkladu.

Husté uložení matice A kromě limitovaného paměťového prostoru neumožňuje plně využít řídkosti matic počítaných systémů. Proto je v současnosti mnoho pozornosti věnováno řídkým implementacím jednotlivých metod. V SPD případě je možné dopředu určit strukturu nenulových prvků faktoru matice A. To nám umožňuje provést nejprve symbolickou a pak numerickou faktorizaci matice soustavy. Tento postup vede k vysoce účinné implementaci Choleského rozkladu, která využívá řídkých datových struktur s nepřímo adresovanými prvky matice A uloženými v jednorozměrném poli. Práce s řídkými maticemi je rozebrána v [18]. Výsledky naší implementace (viz [10], [13]) uvedené v Tabulce 4 ukazují, že CPU časy naměřené na pracovní stanici Crimson jsou výrazně kratší než tytéž údaje

pro počítač Cray. Z výsledků také vyplývá, že v SPD případě u obou počítačů je řídká implementace Choleského metody jasně lepší.

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU		Norma chyby		Norma rezidua	
			Cray	Crimson	Cray	Crimson	Cray	Crimson
1138BUS	1138	2596	0.42	0.07	0.23–09	0.65–12	0.19–05	0.66–08
494BUS	494	1080	0.16	0.03	0.40–10	0.62–12	0.17–09	0.18–11
662BUS	662	1568	0.30	0.05	0.18–09	0.14–13	0.89–10	0.23–12
685BUS	685	1967	0.34	0.05	0.25–10	0.94–13	0.12–09	0.36–11
BRNO974	974	12340	2.30	0.57	0.12–08	0.59–12	0.28–07	0.56–10
GR3030	900	4322	0.79	0.15	0.74–11	0.33–14	0.21–11	0.53–14

Tabulka 4: Výsledky porovnání pro řídkou implementaci Choleského rozkladu.

2.2 Symetrický indefinitní případ.

Pro řešení soustavy lineárních rovnic (2.1) se symetrickou indefinitní maticí jsme uvažovali Bunch-Parlettův rozklad matice $A = RDR^T$, kde D je blokově diagonální matice s bloky rozměrů 1 či 2 a R je součin jednotkové horní trojuhelníkové a permutační matice (viz [3]). Tabulka 5 obsahuje výsledky naměřené pro programy DSIFA a DSISL implementované v souboru programů Linpack, které opět využívají husté datové struktury.

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU		Norma chyby		Norma rezidua	
			Cray	Crimson	Cray	Crimson	Cray	Crimson
BCSSTK09	1083	9760	7.22	10.47	1.21–08	3.98–12	1.16–04	2.98–07
NOS2	957	2547	5.15	5.24	2.28–05	2.70–09	1.42–02	3.05–04
NOS3	960	8402	5.32	6.27	2.78–08	3.67–12	8.87–10	2.02–12
NOS7	729	2673	3.57	5.31	5.21–04	2.75–07	9.58–06	1.36–08
SHERMAN1	1000	3750	5.23	4.18	1.43–09	6.27–13	3.97–12	9.27–15

Tabulka 5: Výsledky porovnání pro hustou implementaci Bunch-Parlettova rozkladu pro symetrické indefinitní matice.

Porovnání časů pro Cray a Crimson ukazuje na to, že indefinitní rozklad je poměrně dobře vektorizovatelný algoritmus. V případě, že chceme využít v tomto algoritmu řídké datové struktury, situace je komplikovanější než pro SPD matici. Nelze totiž provést permutaci systému předem jen za účelem co nejmenšího počtu zpracovávaných nenulových prvků během rozkladu. Permutace též musí přihlížet k skutečným velikostem numerických hodnot, aby rozklad byl stabilní. To klade zvýšené nároky na použité datové struktury. Klasickou implementací řešiče splňující tyto nároky je MA27 (viz [9]). Použití nového grafového modelu je cílem naší implementace, která není dosud dokončena. Proto jsme v experimentech s řídkými systémy se symetrickou indefinitní maticí použili alternativní LU rozklad systému, který je obvykle časově i prostorově náročnější, implementovaný v MA28 (viz [7]). I v tomto řídkém případě jsou výsledky naměřené na pracovní stanici Crimson lepší než na počítači Cray (viz Tabulka 6). Rozdíl proti SPD případu je patrný z porovnání časů pro hustou a řídkou implementaci u jednotlivých počítačů. U obou výsledek silně závisí na struktuře nenulových prvků jednotlivého problému, v některých případech je lepší hustá implementace, jindy zase implementace využívající řídké datové struktury.

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU		Norma chyby		Norma rezidua	
			Cray	Crimson	Cray	Crimson	Cray	Crimson
BCSSTK09	1083	9760	34.91	7.71	0.91–08	0.75–11	0.13–03	0.37–06
BP800	822	4534	2.48	0.45	0.17–09	0.29–11	0.14–09	0.34–12
FS5413	541	4285	2.92	0.59	0.79–06	0.80–09	0.94–06	0.12–08
GRE1107	1107	5664	13.53	3.47	0.17–06	0.29–09	0.19–11	0.58–13
JPWH991	991	6027	11.94	2.65	0.18–09	0.19–13	0.44–10	0.62–13
MAHINDAS	1258	7682	5.72	1.21	0.24–07	0.45–09	0.58–07	0.74–09
MCFE	765	24382	21.44	5.36	0.59–11	0.20–13	0.60+04	0.42+02
NNC666	666	4044	21.05	9.02	0.75–04	0.33–06	0.10–09	0.21–11
NOS2	957	2547	0.33	0.06	0.21–04	0.76–08	0.56–02	0.14–03
NOS3	960	8402	10.51	2.19	0.14–07	0.15–11	0.99–11	0.20–11
NOS7	729	2673	9.71	2.13	0.22–03	0.50–08	0.63–05	0.66–08
ORSIRR1	1030	6858	9.21	1.97	0.51–08	0.41–11	0.19–06	0.34–09
ORSIRR2	886	5970	6.46	1.40	0.54–08	0.60–11	0.16–06	0.35–09
PORES2	1224	9613	33.84	9.49	0.10–07	0.15–11	0.32–05	0.21–07
SHERMAN1	1000	3750	2.47	0.51	0.23–09	0.75–12	0.10–11	0.58–14
WEST0989	989	3537	4.09	0.57	0.26–07	0.35–09	0.73–08	0.21–09

Tabulka 6: Výsledky porovnání pro řídkou implementaci LU rozkladu z MA28 pro symetrické indefinitní matice a pro nesymetrické matice.

2.3 Nesymetrický případ.

Nejrozšířenějším algoritmem pro soustavy s nesymetrickou nesingulární maticí je Gaussova eliminace. Pro naše srovnání jsme uvažovali klasickou Gaussovu eliminaci implementovanou v Linpacku a již zmíněnou řídkou implementaci Gaussovy eliminace MA28. V případe husté implementace (programů DGEFA a DGESL ze souboru Linpack) jsou časy naměřené u počítače Cray lepší než u pracovní stanice Crimson (viz Tabulka 7). Tabulka 6 naproti tomu dokumentuje, že v případě řídké implementace je rychlejší pracovní stanice. Porovnáme-li rychlost husté a řídké implementace pro jednotlivé počítače, vidíme, že u sekvenční pracovní stanice je řídká implementace vždy rychlejší, u počítače Cray je tomu opačně.

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU		Norma chyby		Norma rezidua	
			Cray	Crimson	Cray	Crimson	Cray	Crimson
BP800	822	4534	3.36	3.07	4.73–10	3.42–12	1.53–10	1.29–13
GRE1107	1107	5664	7.06	11.59	1.92–07	4.53–09	9.21–13	1.84–14
JPWH991	991	6027	5.97	10.54	4.79–10	4.11–14	6.44–11	9.06–14
MAHINDAS	1258	7682	7.53	7.94	8.56–09	4.38–10	3.73–08	9.31–10
MCFE	765	24382	3.96	7.58	2.22–11	3.57–13	2.21+03	4.80+01
NNC666	666	4044	2.52	3.14	1.01–04	2.30–06	2.90–10	5.57–12
ORSIRR1	1030	6858	6.23	11.06	6.16–09	6.50–12	2.54–07	6.55–10
ORSIRR2	886	5970	4.62	7.65	5.56–09	5.83–12	2.29–07	5.55–10
PORES2	1224	9613	12.74	42.23	8.25–09	4.71–10	1.74–06	3.01–08
WEST0989	989	3537	4.82	4.39	7.53–07	1.86–08	4.61–09	6.18–11

Tabulka 7: Výsledky porovnání pro hustou implementaci LU rozkladu pro nesymetrické matice.

2.4 Implicitní Gaussova eliminace.

Implementace SIG implicitní Gaussovy eliminace pro řídké lineární systémy byla popsána v [16]. Její důležité vlastnosti jsou následující: SIG poskytuje výrazně přesnější reziduum systému než klasická Gaussova eliminace, poskytuje obvykle mírně přesnější řešení než klasická Gaussova eliminace,

implicitní Gaussova eliminace je mnohem lépe vektorizovatelná, je obvykle pomalejší než Gaussova eliminace z důvodu většího zaplnění novými nenulovými prvky.

Řešení řídkých lineárních systémů má většinou do plného využití vektorizace daleko. Zvolený algoritmus však umožňuje použít řídké, husté či kombinované uložení pomocných matic ve kterých se uchovává zaplnění, a které jsou nazývány matice deflace. To implikuje různé stupně vektorizovatelnosti algoritmu. Algoritmus nemá typický zpětný chod, neboť řešení systému je iteračně upravováno v průběhu rozkladu a není zapotřebí uchovávat faktory.

Datové struktury pro matici A využívají vždy její řídkosti. Pro uložení matic matic deflace je použito buď řídkých nebo úplně hustých struktur. Tři způsoby použití algoritmu se liší parametrem σ a odpovídají různým způsobům kombinace řídkých a hustých struktur pro uložení matic deflace.

Přepnutí do hustých struktur je jedna z důležitých cest pro urychlení řídké Gaussovy eliminace. To platí dvojnásob při použití vektorového počítače. Tato technika znamená, že matice je v určitém okamžiku běhu algoritmu přepsána do hustých struktur a zbytek algoritmu probíhá jako obyčejná hustá Gaussova eliminace. To má význam pro skalární výpočty, ale především pro vektorové výpočty, neboť matice je obvykle v pokročilejší fázi eliminace hustá a navíc dostatečně malá, aby se vešla do operační paměti. Problémem je obvykle určit okamžik přepnutí. V případě implicitní Gaussovy eliminace je tento okamžik možné určit snáze a navíc jeho volba není zdaleka tak kritická, jak bylo ukázáno v [16]. V našich experimentech jsme použili tři hodnoty $\sigma : 0, \frac{4}{5}n, n$. První z nich odpovídá algoritmu s maticemi deflace uloženými v úplně hustých datových strukturách (viz Tabulka 10), druhý z nich přepíná v zdánlivě ideálním okamžiku (viz [16], Tabulka 9). Hodnota $\sigma = n$ odpovídá úplně řídké implementaci implicitní Gaussovy eliminace (viz Tabulka 8).

V případě **řídkých** datových struktur jsou operace s nepřímo adresovanými vektory, které se používají v řídké implementaci, vektorizovatelné pouze s pomocí technického zařízení "gather-scatter" dostupném na počítačích Cray Y-MP. Asymptotický výkon tohoto počítače pro cykly s nepřímo adresovanými vektory je zhruba poloviční proti týmž cyklům s přímo adresovanými vektory. Přesto však práce s těmito nepřímo adresovanými vektory může být neefektivní vzhledem k malému počtu nenulových prvků v řádcích (tj. malé délce vektorů) v některých maticích.

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU		Norma chyby		Norma rezidua	
			Cray	Crimson	Cray	Crimson	Cray	Crimson
BP800	822	4534	1.13	0.36	0.23–09	0.35–11	0.30–10	0.41–12
FS5413	541	4285	60.12	22.64	0.37–07	0.24–08	0.45–07	0.30–08
GRE1107	1107	5664	160.88	40.25	0.20–05	0.35–07	0.18–10	0.41–12
JPWH991	991	6027	>200	64.40	–	0.79–11	–	0.12–10
MAHINDAS	1258	7682	11.67	3.42	0.19–06	0.34–08	0.52–07	0.10–07
MCFE	765	24382	123.85	38.46	0.13–09	0.27–11	0.99+04	0.15+03
NNC666	666	4044	61.62	16.28	0.83–04	0.13–05	0.21–09	0.56–11
ORSIRR1	1030	6858	131.35	51.58	0.40–08	0.32–10	0.26–06	0.54–08
ORSIRR2	886	5970	61.41	18.28	0.14–08	0.46–10	0.19–06	0.51–08
PORES2	1224	9613	170.52	62.64	0.31–08	0.15–08	0.82–04	0.27–05
WEST0989	989	3537	3.00	0.77	0.15–07	0.19–09	0.18–07	0.17–08

Tabulka 8: Výsledky porovnání pro implementaci implicitní Gaussovy eliminace pro nesymetrické matice s řídkými maticemi deflace.

Pro **husté** datové struktury je celá aktualizace matic deflace velmi dobře vektorizovatelná. Část programu používající husté datové struktury byla přímo napsána s úmyslem dobré vektorizace. Zbytek kódu byl vektorizován pouze kompilátorem - t.j. nebyly deklarovány datové nezávislosti, optimalizovány krátké smyčky, ...

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU		Norma chyby		Norma rezidua	
			Cray	Crimson	Cray	Crimson	Cray	Crimson
BP800	822	4534	0.59	0.90	0.29–09	0.30–11	0.15–10	0.35–12
FS5413	541	4285	41.46	17.98	0.29–07	0.11–08	0.41–07	0.23–08
GRE1107	1107	5664	43.71	17.23	0.82–06	0.27–07	0.60–11	0.24–12
JPWH991	991	6027	123.56	42.19	0.88–10	0.55–11	0.70–10	0.78–11
MAHINDAS	1258	7682	1.40	3.39	0.21–07	0.18–09	0.15–07	0.31–08
MCFE	765	24382	75.86	26.23	0.13–09	0.15–11	0.10+05	0.15+03
NNC666	666	4044	37.62	10.96	0.64–04	0.84–06	0.26–09	0.48–11
ORSIRR1	1030	6858	53.82	25.86	0.11–08	0.20–10	0.58–07	0.31–08
ORSIRR2	886	5970	23.77	10.55	0.10–08	0.43–10	0.18–06	0.42–08
PORES2	1224	9613	108.88	49.00	0.11–07	0.34–09	0.12–04	0.19–05
WEST0989	989	3537	0.67	0.87	0.31–07	0.17–09	0.34–07	0.56–09

Tabulka 9: Výsledky porovnání pro implementaci implicitní Gaussovy eliminace pro nesymetrické matice s kombinovanými datovými strukturami pro matice deflace.

Pro řidší matice (BP, MAHINDAS, WEST) je obvykle řídká implementace rychlejší než hustá implementace na obou počítačích. To je dáno jednoduše proto, že v druhém případě máme výrazně větší absolutní počet operací. Vektorizace na počítači Cray se velmi osvědčuje, neboť zmenšuje poměr časů dosažených pro $\sigma = 0$ a $\sigma = n$. Časy dosažené pro $\sigma = \frac{4}{5}n$ jsou pro řidší matice často optimální, obzvláště pro počítač Cray, vzhledem k efektivní vektorizaci hustých datových struktur. Pro matice trpící větším zaplněním mohou být na obou počítačích rychlejší časy husté implementace. Vzhledem k vektorizaci jsou tyto časy pro Cray často i absolutně lepší.

Globální pohled na Tabulky 8 – 10 v absolutních časech však ukazuje převahu pracovní stanice pro řídké úlohy a nepříliš kratší časy počítače Cray i v případě poměrně dobře vektorizovatelné úlohy pro $\sigma = 0$. Výrazně menší časy pro Cray oproti výsledkům pro SGI jsou pozorovatelné pouze pro úlohy nabírající velké zaplnění nenulovými prvky od počátku algoritmu (ORSIRR1, ORSIRR2).

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU		Norma chyby		Norma rezidua	
			Cray	Crimson	Cray	Crimson	Cray	Crimson
BP800	822	4534	4.52	3.09	0.14–09	0.26–11	0.13–10	0.33–12
FS5413	541	4285	2.38	6.61	0.42–07	0.53–09	0.37–07	0.14–08
GRE1107	1107	5664	17.09	63.27	0.82–08	0.14–08	0.13–11	0.29–13
JPWH991	991	6027	9.54	23.93	0.21–09	0.44–13	0.34–10	0.62–13
MAHINDAS	1258	7682	10.88	9.24	0.21–07	0.12–09	0.13–06	0.82–08
MCFE	765	24382	5.94	12.61	0.58–11	0.26–13	0.22+04	0.55+02
NNC666	666	4044	4.25	7.80	0.55–04	0.31–05	0.23–09	0.36–11
ORSIRR1	1030	6858	8.42	24.76	0.13–08	0.24–11	0.44–07	0.66–09
ORSIRR2	886	5970	6.86	16.17	0.11–08	0.13–11	0.37–07	0.58–09
PORES2	1224	9613	15.16	40.38	0.14–07	0.42–11	0.37–05	0.68–07
WEST0989	989	3537	6.39	4.82	0.23–07	0.42–09	0.35–07	0.33–09

Tabulka 10: Výsledky porovnání pro implementaci implicitní Gaussovy eliminace pro nesymetrické matice s hustými maticemi deflace.

3 Řešení symetrického problému vlastních čísel a problému singulárních čísel.

Úplný problém vlastních čísel symetrické matice $A \in R^{N,N}$ je faktorizace $A = Q\Lambda Q^T$, kde Q je ortogonální matice ($Q^TQ = QQ^T = I$) a $\Lambda = diag(\lambda_1, \ldots, \lambda_N)$ je diagonální matice s reálnými

prvky. Standardním algoritmem pro řešení úplného problému vlastních čísel je redukce symetrické matice na reálnou tridiagonální formu a v druhém kroku výpočet vlastních čísel tridiagonální matice pomocí varianty QR algoritmu (viz [12], [1]). Pro naše porovnání jsme použili program DSYEV ze souboru Lapack, který využívá husté uložení matice v dvourozměrném poli. Výsledky uvedené v Tabulce 11, podobně jako u ostatních hustých algoritmů, ukazují na převahu počítače Cray pro snadno vektorizovatelné úlohy.

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU	
			Cray	Crimson
BCSSTK09	1083	9760	43.10	163.35
NOS2	957	2547	28.63	109.39
NOS3	960	8402	32.77	120.21
NOS7	729	2673	15.58	48.53
SHERMAN1	1000	3750	32.09	110.94
SHERMAN2	1080	23094	41.47	170.31
SHERMAN4	1104	3786	36.44	127.96

Tabulka 11: Výsledky porovnání pro symetrický problém vlastních čísel.

Problém singulárních čísel obecné (v našem případě nesingulární nesymetrické) matice A je rozklad $A = U\Sigma V^T$, kde U a V jsou ortogonální matice a Σ diagonální matice s nezápornými diagonálními prvky. Nejznámějším postupem pro řešení tohoto problému je algoritmus, který v prvním kroku redukuje matici A na horní bidiagonální formu a v druhém kroku pomocí varianty QR algoritmu počítá singulární čísla bidiagonální matice B (viz podrobnější popis např. v [15] a [6]). Pro naše porovnání jsme použili program DSVDC z programového souboru Linpack, který opět využívá husté datové struktury pro matici A. I v tomto případě výsledky uvedené v Tabulce 12 ukazují, že algoritmus je dobře vektorizovatelný a časy naměřené na počítači Cray jsou lepší v porovnání s pracovní stanicí.

MATICE	Rozměr N	P. nenul. prvků NZ	Čas CPU	
			Cray	Crimson
BP800	822	4534	72.38	220.31
GRE1107	1107	5664	181.08	559.89
JPWH991	991	6027	126.84	399.98
MAHINDAS	1258	7682	>200	796.48
MCFE	765	24382	42.93	155.95
NNC666	666	4044	38.46	112.09
ORSIRR1	1030	6858	116.85	440.69
ORSIRR2	886	5970	81.51	276.54
PORES2	1224	9613	157.16	746.98
WEST0989	989	3537	88.62	383.75

Tabulka 12: Výsledky porovnání pro problém singulárních čísel

4 Závěr.

Dosud uvedené porovnání neobsahuje hodnocení velikostí normy chyby $||\bar{x} - x||$ a normy rezidua $||b - A\bar{x}||$. Tyto údaje zahrnuté ve všech tabulkách dokumentují známé výrazně horší vlastnosti akumulace chyb u počítačů Cray řady Y-MP (viz [4]). Velikosti normy chyb a reziduí jsou na pracovní stanici v průměru o jeden až dva řády lepší než u počítače Cray. Odpovídá to přibližně teoretickým vlastnostem aritmetik uvedených v úvodu.

Dále jsme porovnávali rychlost počítačů na lineárních úlohách využívajících husté a řídké datové struktury. U hustých algoritmů jsme převážně zaznamenali lepší CPU časy u počítače Cray, který využíval vysoký stupeň jejich vektorizovatelnosti. Tento přístup ale naráží na problém paměťových omezení. Rozměry použitých matic odpovídají přibližně paměťovým možnostem obou počítačů, v případě vyšších rozměrů by byl výpočet v hustých strukturách prakticky nemožný, případně by se podstatně odrazil na reálné průchodnosti počítačů.

V porovnání jsme použili některé úlohy lineární algebry, které patří k dobře vektorizovatelným úlohám. Řídké datové struktury, které jsou moderní algoritmy nuceny používat, ovšem obsahují nepřímé adresování prvků matic a tím se možnosti vektorizace dále snižují. Výsledky ukazují, že v těchto případech jsou časy dosažené na pracovních stanicích lepší.

Výsledky testů nevyznívají z výše uvedených důvodů pro Cray Y-MP EL příznivě. Tento počítač není určen k počítání vědecko-technických úloh, ale spíše k přípravě dat pro kompatibilní výkonné členy řady Y-MP. V neposlední řadě je nutné připomenout další důležitý ukazatel při hodnocení počítačů, poměr cena/výkon, který je u pracovní stanice mnohem příznivější.

5 Poděkování.

Autoři děkují ing. V. Šebestovi, DrSc. za cenné připomínky k tomuto článku. Naše poděkování patří také pracovníkům výpočetního střediska FzÚ AV ČR za bezplatné umožnění těchto testů v rámci grantu GA AV ČR č. 23003 a grantu GA ČR č. 201/93/0067. Děkujeme též panu Karlu Hnízdilovi za pomoc při přípravě rukopisu a tabulek.

Literatura

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J.J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen: LAPACK User's guide, SIAM, Philadelphia, 1992.

[2] W.J. Blume: Success and Limitations in Automatic Parallelization of the Perfect Benchmarks Programs, CSRD Report No. 1249. University of Illinois, 1992.

[3] J.R. Bunch, B.N. Parlett: Direct methods for solving symmetric indefinite systems of linear equations, SIAM J. Numer. Anal. 8(1971), pp. 639-655.

[4] J.W. Demmel: Trading off Parallelism and Numerical Stability, in: Linear Algebra for Large-Scale and Real-Time Applications, M.S.Moonen et al., eds., Kluwer Academic Publishers, pp. 49–68, 1993.

[5] J.J. Dongarra: Performance of Various Computers Using Standard Linear Equations Software, Technical Report CS-89-85, University of Tennessee, Update 1.2.1993.

[6] J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart: LINPACK User's Guide, SIAM, Philadelphia, 1979.

[7] I.S. Duff: MA28 — a Set of FORTRAN Subroutines for Sparse Unsymmetric Linear Equations, AERE Harwell, R8730, Her Majesty's Stationery Office, London, 1977.

[8] I.S. Duff, R.G. Grimes, J.G. Lewis: Sparse matrix test problems, ACM Trans. Math. Software 15 (1989), pp. 1-14.

[9] I.S. Duff, J.K. Reid: MA27 – A set of FORTRAN subroutines for solving sparse symmetric sets of linear equations, Tech. Rep. AERE R 10533, Harwell Laboratory, Oxfordshire, 1982.

[10] A. George, J.W.H. Liu: Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, N.J., 1981.

[11] G.H. Golub, C. van Loan: Matrix Computations, Johns Hopkins University Press, Baltimore, MD, 2nd edition, 1989.

[12] A. Greenbaum, J.J. Dongarra: Experiments with QL/QR methods for the symmetric tridiagonal eigenproblem, Technical Report CS-89-92, University of Tennessee, Knoxville, 1989.

[13] J.W.H. Liu: The role of elimination trees in sparse factorization, SIAM J. Matrix Anal. Appl. 11 (1990), 134-172.

[14] L. Pointer: Perfect: Performance Evaluation for Cost-effective Transformations. Report 2., CSRD Report No. 964, University of Illinois, 1990.

[15] G.W. Stewart: On the perturbation of pseudo-inverses, projections, and linear least squares problems, SIAM Review 4(1977), 634-662.

[16] M. Tůma: Solving Sparse Unsymmetric Sets of Linear Equations Based on Implicit Gauss Projection, Technical Report No. 556, UIVT AV ČR, 1993.

[17] J.H. Wilkinson: Rounding Errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, N.J., 1963.

[18] Z. Zlatev: Computational Methods for General Sparse Matrices, Kluwer Academic Publishers, Dordrecht, 1991.