Parallele Programmierung - Solitaire Chess

Kira Duwe - Enno Zickler

DKRZ- UHH

7. Oktober 2013

Spielregeln

- Figuren ziehen nach Schachregeln
- pro Zug eine Figur ziehen, um eine andere zu schlagen
- 4 x 4 - Brett
- 10 Figuren : 2 x Bauer, Türme, Springer, Läufer; 1 x König und Dame
- Ziel: eine einzige Figur bleibt übrig

Fragestellungen

Vorhaben

- Welche Startpositionen sind lösbar? (mit variierender Figurenanzahl)
- Mögliche Erweiterungen:
  - größeres Spielbrett
  - andere Brettform
  - In wie viel Zügen ist ein Spiel lösbar?
  - Wie viele unterschiedliche zum Ziel führende Zugmöglichkeiten gibt es?

Ergebnis

- Wie viele Spielbretter sind lösbar
- Alle Spielbretter mit 1 bis 10 Figuren
- Variable Brettgröße von 1 bis 21 Felder in beliebiger rechteckiger Form

Spielbrettdarstellung

Oktaldarstellung

- 16 Felder * 3 bit = 48 bit
- uint64 ist ausreichend
- kleiner als Array (16 * 8 bit = 128 bit)

0	1	0	0
0	5	6	0
4	2	1	4
2	3	0	3

Tabelle: Spielbrett 3032 4124 0650 0010

Figuren

- 0 = leeres Feld
- 1 = Bauer
- 2 = Turm
- 3 = Läufer
- 4 = Springer
- 5 = König
- 6 = Dame

Spielbretterzeugung

- Verschachtelte for-Schleifen, für jede Figur von 0 bis Spielbrettgröße
- bei doppelten Figuren sollte die 2. Figur abhängig von der 1. sein
- Abschneiden der Schleifendurchläufe, wenn betrachtetes Feld nicht frei

Spielbretterzeugung I

```
for(Dame von 0 bis 16)
  if( Feldfrei)
    for(Koenig von 0 bis 16)
      if( Feldfrei)
        for (Springer1 von 0 bis 16)
          if( Feldfrei)
            for (Springer2 von posSpringer1 bis
               ↪ 16)
              ...
                for (Bauer2 von posBauer1 bis 16)
                  Spielbrettberechnen
```

Spielbrettberechnung

- Dynamische Programmierung
- Vorherige Lösungen werden wieder verwendet
- Erzeugung nach Figurenanzahl (+ for-Schleife)
- Berechnung nach Figurenanzahl aufsteigend
- Lösungen der vorherigen Ebene müssen bekannt sein

Spielbrettberechnung

- Umwandlung in Array für Zugberechnung
- Felderweise Überprüfung des gesamten Brettes, ob durch möglichen Zug ein lösbares Brett entsteht
- Zugriff auf vorherige Lösungen
- Abbruch der Berechnung, wenn Nachfolgebrett als lösbar gespeichert

Schlagen der Figuren I

```
einser_Bitmaske = 0xffffffffffffffffLL;

// Spielfiguren, geschlagene und schlagende, von
   ↪ Spielbrett loeschen
// Von der Bitmaske wird "(7 << pos*3)"
   ↪ abgezogen, um an dieser Stelle 0 zu erzeugen

  neues_spielbrett = spielbrett &
  (einser_Bitmaske - (7 << pos*3) - (7 <<
     ↪ neue_pos*3));

// nach Schlagen Spielfigur neu  setzen
  neues_spielbrett +=
  (DarstellungFigur << neue_pos*3);
```

Größe des Problems

- 6.7 Milliarden Spielbretter für 4x4
- sehr großer Anteil lösbar

Figuren:	Anzahl	% Lösbar:
1	96	100.00
2	4.080	50.25
3	100.800	70.95
4	1.594.320	89.41
5	16.773.120	97.62
6	118.198.080	99.70
7	547.747.200	99.98
8	1.589.187.600	100.00
9	2.594.592.000	100.00
10	1.816.214.400	100.00
Gesamt	6.684.411.696	99.98

Speicherung

- Abspeichern der Spielbretter in Hashtablle / Hashset
- 6.684.411.696 x 64bit x 2 = 855.604.697.088 bit = 106 GByte
- Hashset fürt zu Halbierung des Speicherbedarfs
- Optimierung durch speichern der nicht lösbaren
- 1.047.721 x 64 bit = 67.054.144 bit = 67 MB

Output Solitaire-Schach 4x4 auf 8 Knoten mit je 24 Threads

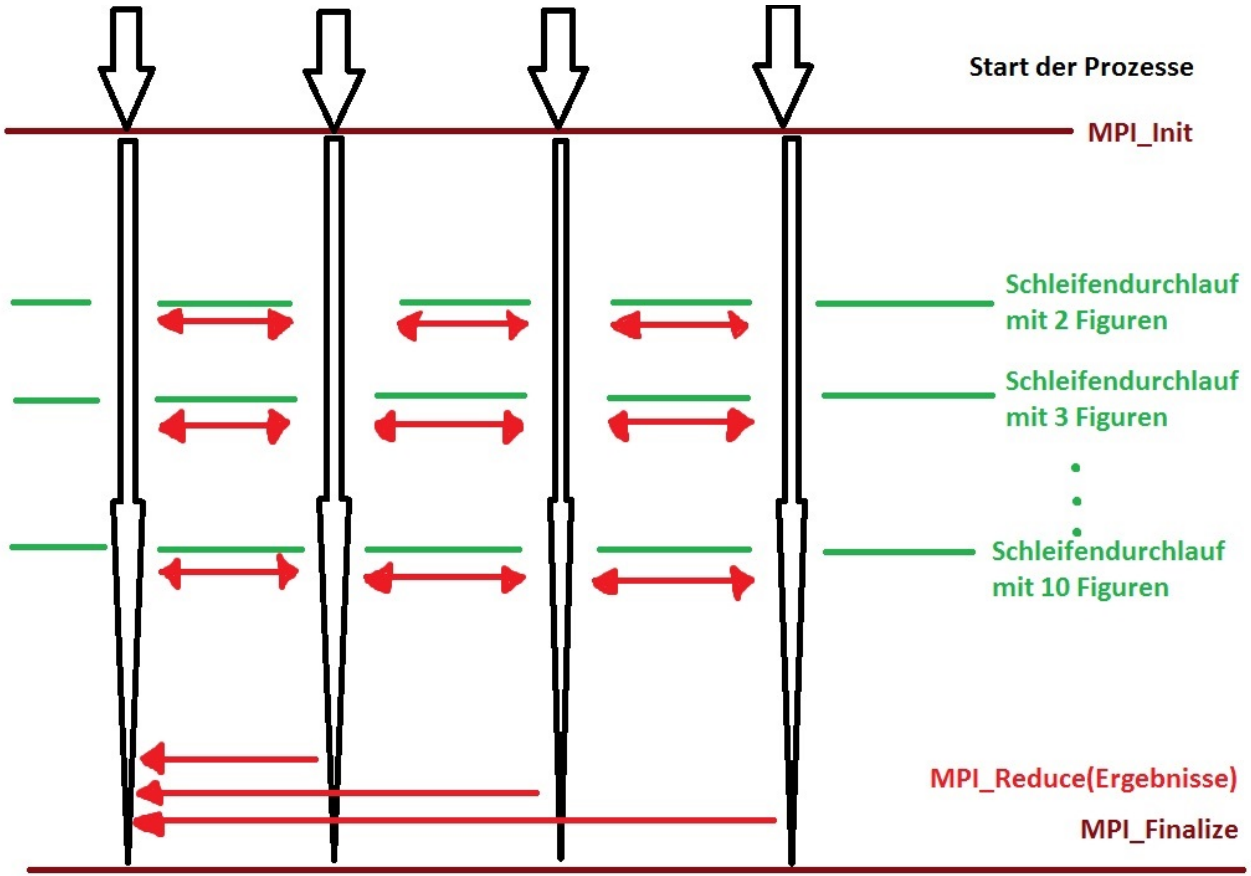
```
Berechnungszeit:    191.835981 s
Bretter / Sekunde: 34844430.148899

Figuren:  Loesbare:    Gesamt:  % Lösbar:  Zeit Max    Min     Avg
=============================================================================
  0              0          0      -nan      0.00     0.00     0.00
  1             96         96   100.00      0.00     0.00     0.00
  2           4482       7920    56.59      0.02     0.01     0.01
  3          71521     100800    70.95      0.05     0.02     0.03
  4        1425531    1594320    89.41      1.58     0.18     0.56
  5       16373752   16773120    97.62      9.25     4.41     5.74
  6      117846720  118198080    99.70     28.10    13.09    16.25
  7      547652219  547747200    99.98     59.50    39.19    43.58
  8     1589184012 1589187600   100.00     61.96    47.42    59.09
  9     2594592000 2594592000   100.00     46.06    25.67    40.58
 10     1816214400 1816214400   100.00     22.31     8.35    19.83
=============================================================================
Summe:  6683363975 6684414768    99.98    191.84
```

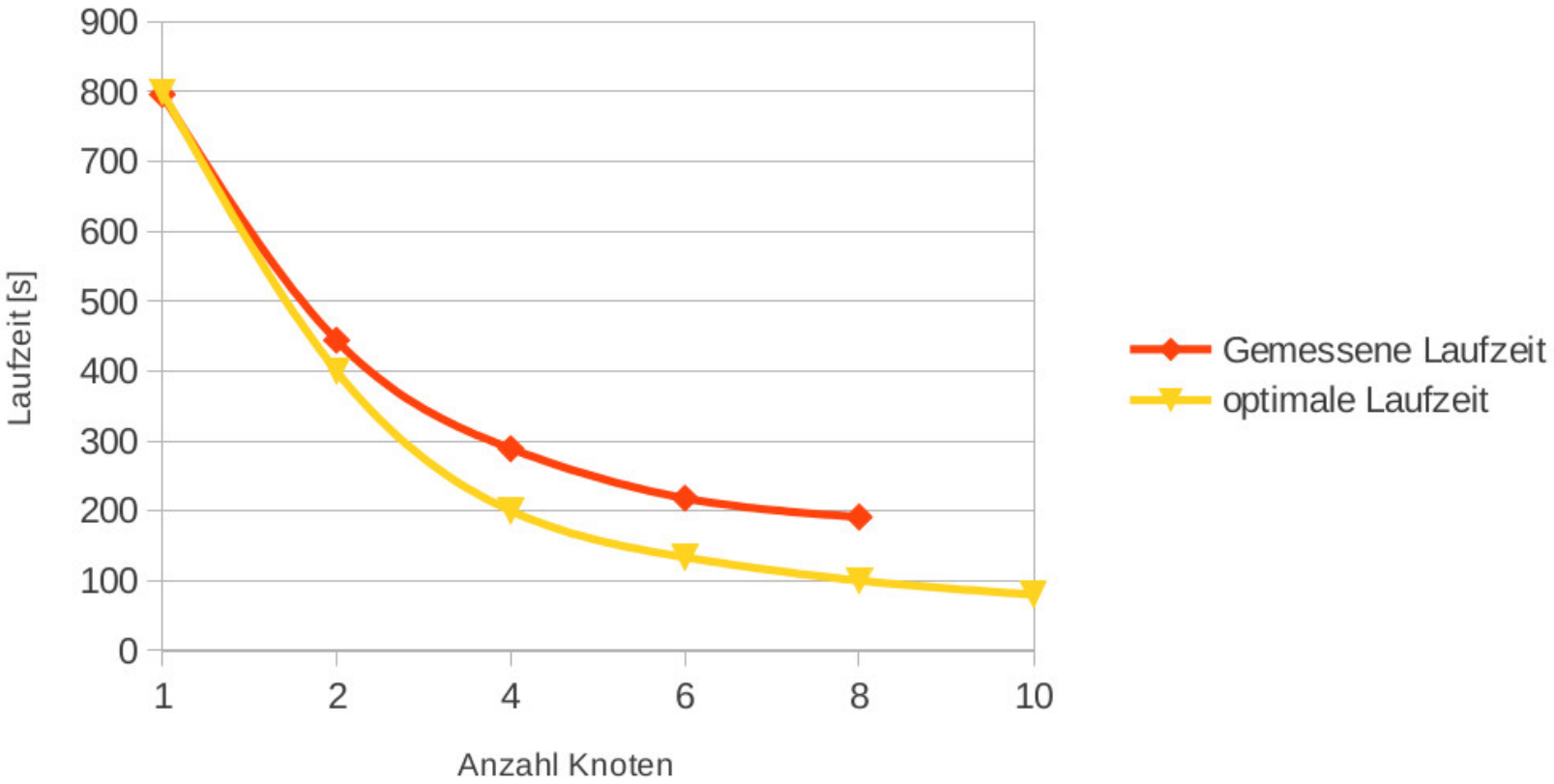
gprof

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 32.51     11.75    11.75 181063344     0.00     0.00  spielbrettBerechne
 28.54     22.06    10.31        1    10.31    36.02  spielbretter_berechne
 11.15     26.08     4.03 1606358325     0.00     0.00  feldFrei
  8.52     29.16     3.08 181063344     0.00     0.00  spielbretterArrayCreate
  5.76     31.24     2.08 181063344     0.00     0.00  spielbretterArrayDestruct
  3.84     32.63     1.39 183252731     0.00     0.00  schlageFigur
  1.70     33.24     0.62 39687589      0.00     0.00  berechneSpringer
  1.56     33.81     0.57 39762256      0.00     0.00  berechneLaeufer
  1.54     34.36     0.56 39927693      0.00     0.00  berechneTurm
  1.52     34.91     0.55 183252731     0.00     0.00  neuesSpielbrettLoesbar
  1.15     35.33     0.42 39052394      0.00     0.00  berechneBauer
  1.02     35.70     0.37 20782134      0.00     0.00  berechneKoenig
  0.86     36.01     0.31 20732220      0.00     0.00  berechneDame
  0.26     36.10     0.10                              main
  0.07     36.13     0.03                              frame_dummy
  0.04     36.14     0.02        1     0.02     0.02  erzeugeHashtables
  0.00     36.14     0.00        1     0.00     0.00  AskParams
  0.00     36.14     0.00        1     0.00     0.00  gibStatistikAus
  0.00     36.14     0.00        1     0.00     0.00  spielbretterErzeugung1Figur
```

MPI-Kommunikation



Speedup 4x4-Spielbrett mit 24 Thread/Knoten



Vampirtrace - Profile

*excl. time	incl. time	calls	excl. time / call	incl. time / call	name
0.601s	0.601s	19.92	30.201ms	30.201ms	!$omp ibarrier @spielbretter.c:545
40.338ms	0.619s	20.92	1.929ms	29.617ms	!$omp for @spielbretter.c:329
26.816ms	26.816ms	0.23	0.117s	0.117s	MPI_Barrier
18.551ms	3.010ms	7560.33	2.454us	0.398us	spielbretterArrayCreate
11.405ms	1.138ms	8549.56	1.334us	0.133us	schlageFigur
10.480ms	2.200ms	7559.96	1.386us	0.291us	spielbretterArrayDestruct
9.443ms	4.904ms	37300.17	0.253us	0.131us	feldFrei
9.059ms	2.196ms	2110.31	4.292us	1.040us	berechneDame
8.904ms	19.387ms	7560.35	1.177us	2.564us	spielbrettBerechne
7.608ms	1.708ms	1986.56	3.830us	0.860us	berechneSpringer
7.172ms	7.172ms	0.92	7.824ms	7.824ms	MPI_Bcast
3.602ms	3.602ms	8549.42	0.421us	0.421us	neuesSpielbrettLoesbar
2.355ms	58.659ms	1.23	1.916ms	47.722ms	parallel region
1.986ms	1.199ms	1347.17	1.474us	0.890us	berechneTurm
1.942ms	1.008ms	1466.42	1.324us	0.687us	berechneLaeufer
1.782ms	0.802ms	1410.73	1.263us	0.568us	berechneBauer
0.757ms	0.837ms	0.04	18.172ms	20.093ms	MPI_Init
0.641ms	0.000ns	0.04	15.373ms	0.000ns	spielbretter_berechne
0.565ms	1.272ms	1378.92	0.409us	0.922us	berechneKoenig
0.251ms	0.000ns	0.04	6.030ms	0.000ns	user
0.166ms	0.266ms	259.21	0.640us	1.027us	!$omp critical @spielbretter.c:209
0.100ms	0.100ms	259.21	0.387us	0.387us	!$omp critical sblock @spielbretter.c:210
80.032us	80.032us	0.04	1.921ms	1.921ms	sync time
19.662us	0.620s	20.92	0.940us	29.618ms	!$omp parallel @spielbretter.c:329
3.786us	0.000ns	0.04	90.873us	0.000ns	main
2.256us	2.256us	0.04	54.148us	54.148us	erzeugeHashtables
0.661us	0.661us	0.04	15.856us	15.856us	AskParams
0.199us	0.199us	0.08	2.394us	2.394us	MPI_Initialized
75.743ns	75.743ns	0.04	1.818us	1.818us	spielbretterErzeugung1Figur
38.246ns	38.246ns	0.04	0.918us	0.918us	MPI_Comm_size
12.374ns	12.374ns	0.04	0.297us	0.297us	MPI_Comm_rank
0.000ns	0.000ns	1	0.000ns	0.000ns	tracing off