

Effiziente Programme

F. Gruber, P. Hönisch, C. Hochreiner, M. Petritsch

21. 1. 2011

- Prozessor: i7
- Testprogramm: papiex
- Testscript: testet gleichzeitig auf o0 und o3
- Versionskontrolle: git

Replaced computation with temp

```
src/shortest-path.c

...  ... @@ -3837,12 +3837,13 @@ static void prepare_super_table()
3837 3837     ss->next = *ss_listp;
3838 3838     *ss_listp = ss;
3839 3839     } else {
3840 - int hash = hash_super(super2+c->offset, c->length);
3840 + int tmp = super2 + c->offset;
3841 + int hash = hash_super(tmp, c->length);
3841 3842     struct super_table_entry **p = &super_table[hash];
3842 3843     struct super_table_entry *e = malloc(sizeof(struct super_table_entry));
3843 3844     ss->next = NULL;
3844 3845     e->next = *p;
3845 - e->start = super2 + c->offset;
3846 + e->start = tmp;
3846 3847     e->length = c->length;
3847 3848     e->ss_list = ss;
3848 3849     *p = e;
```

Result

Inline method

```
3972 - init_waypoints(inst[ninsts]);
3972 + //init_waypoints();
3973 + int k;
3974 +
3975 + for (k=0; k<MAX_STATE; k++)
3976 +     inst[ninsts][k].cost=INF_COST;
3977 +
3978 inst[ninsts][CANONICAL_STATE].cost=0;
3979 transitions(inst[ninsts],trans[ninsts]);
3980 for (i=ninsts-1; i>=0; i--) {
3981 - init_waypoints(inst[i]);
3981 + //init_waypoints(inst[i]);
3982 + int k;
3983 + for (k=0; k<MAX_STATE; k++)
3984 +     inst[i][k].cost=INF_COST;
3985 +
```

Result

Make global constants makros



src/shortest-path.c

View f

```
... @@ -456,11 +456,9 @@ N_lit_execute,
456 456     N_START_SUPER
457 457 } PrimNum;
458 458
459 -static int no_dynamic=0; /* if true, no code is generated
460 -    dynamically */
461 -static int static_super_number = 10000; /* number of ss used if available */
462 -#define MAX_STATE 9 /* maximum number of states */
463 -static int maxstates = MAX_STATE; /* number of states for stack caching */
459 +#define NO_DYNAMIC        0 /* if true, no code is generated dynamically */
460 +#define STATIC_SUPER_NUMBER 10000 /* number of ss used if available */
461 +#define MAX_STATE        9 /* maximum number of states */
464 462
465 463 FILE *output;
466 464
... @@ -2085,7 +2083,7 @@ const char const* const prim_names[]={
2085 2083
2086 2084 static int is_relocatable(int p)
2087 2085 {
2088     return !no_dynamic && priminfos[p].start != NULL;
2086 + return !NO_DYNAMIC && priminfos[p].start != NULL;
2089 2087 }
```

Make global constants makros

```
...    ... @@ -3823,8 +3821,8 @@ static void prepare_super_table()
3823 3821
3824 3822     for (i=0; i<sizeof(super_costs)/sizeof(super_costs[0]); i++) {
3825 3823         struct cost *c = &super_costs[i];
3826 -         if ((c->length < 2 || nsupers < static_super_number) &&
3827 -             c->state_in < maxstates && c->state_out < maxstates) {
3824 +         if ((c->length < 2 || nsupers < STATIC_SUPER_NUMBER) &&
3825 +             c->state_in < MAX_STATE && c->state_out < MAX_STATE) {
3828 3826             struct super_state **ss_listp= lookup_super(super2+c->offset, c->length);
3829 3827             struct super_state *ss = malloc(sizeof(struct super_state));
3830 3828             ss->super= i;
...    ... @@ -3912,7 +3910,7 @@ void init_waypoints(struct waypoint ws[])
3912 3910     {
3913 3911         int k;
3914 3912
3915 -         for (k=0; k<maxstates; k++)
3913 +         for (k=0; k<MAX_STATE; k++)
3916 3914             ws[k].cost=INF_COST;
3917 3915     }
3918 3916
...    ... @@ -3921,7 +3919,7 @@ void transitions(struct waypoint inst[], struct waypoint trans[])
3921 3919     int k;
3922 3920     struct super_state *l;
3923 3921
3924 -     for (k=0; k<maxstates; k++) {
3922 +     for (k=0; k<MAX_STATE; k++) {
3925 3923         trans[k] = inst[k];
3926 3924         trans[k].no_transition = 1;
3927 3925     }
```


Result

Removed function pointer

```
src/shortest-path.c | View
...  ... @@ -3933,7 +3933,7 @@ void transitions(struct waypoint inst[], struct waypoint trans[])
3933 3933     struct waypoint *wo=&(inst[c->state_out]);
3934 3934     if (wo->cost == INF_COST)
3935 3935         continue;
3936 - jcost = wo->cost + ss_cost(s);
3936 + jcost = wo->cost + cost_codesize(s);
3937 3937     if (jcost <= wi->cost) {
3938 3938         wi->cost = jcost;
3939 3939         wi->inst = s;
...  ... @@ -3999,7 +3999,7 @@ void optimize_rewrite(PrimNum origs[], int ninsts)
3999 3999     }
4000 4000     if (wo->cost == INF_COST)
4001 4001         continue;
4002 - jcost = wo->cost + ss_cost(s);
4002 + jcost = wo->cost + cost_codesize(s);
4003 4003     if (jcost <= wi->cost) {
4004 4004         wi->cost = jcost;
4005 4005         wi->inst = s;
```

Result

Result

Pointer Arithmetik

```
4051 - PrimNum *start = data;
4052 4059 size_t input_size;
4053 - int i;
4054 4060
4055 4061 prepare_super_table();
4056 4062 input_size = fread(data, sizeof(PrimNum), MAX_INPUT_SIZE, stdin);
4057 - for (i = 0; i < input_size; i++)
4058     - if (data[i] == -1) {
4059     -     optimize_rewrite(start, data+i-start);
4060     -     start = data+i+1;
4063 +
4064 + PrimNum *start = data;
4065 + PrimNum *end = data + input_size;
4066 +
4067 + for ( PrimNum *pn = data; pn != end; pn++ ) {
4068 +     if ( *pn == -1 ) {
4069 +         optimize_rewrite( start, pn - start );
4070 +         start = pn + 1;
4061 4071     }
4072 + }
```

Result

Loop indices

```
...  ... @@ -3849,12 +3849,11 @@ struct waypoint {
3849 3849
3850 3850 void transitions(struct waypoint inst[], struct waypoint trans[])
3851 3851 {
3852 3852     - int k;
3853 3853     -
3854 3854     - for (k=0; k<MAX_STATE; k++) {
3852 3852 + for ( int k=MAX_STATE - 1; k>= 0; k--) {
3855 3853     trans[k] = inst[k];
3856 3854     trans[k].no_transition = 1;
3857 3855     }
3856 3856 +
3858 3857     const PrimNum*      start = state_transitions;
3859 3858     const PrimNum* const end  = state_transitions + ARRAY_LEN( state_transitions );
3860 3859
...  ... @@ -3902,12 +3901,11 @@ void optimize_rewrite(PrimNum origs[], int ninsts)
3902 3901     int nextdyn, nextstate, no_transition;
3903 3902
3904 3903     //init_waypoints();
3905 3905     - int k;
3906 3906     -
3907 3907     - for (k=0; k<MAX_STATE; k++)
3904 3904 +// struct waypoint* wp = inst[ninsts];
3905 3905 + for ( int k=MAX_STATE - 1; k>0; k--)
3908 3906     inst[ninsts][k].cost=INF_COST;
```

Result

Basic blocks

```
...    ... @@ -4003,15 +4007,30 @@ int main(int argc, char **argv, char **env)
4003 4007 // prepare_super_table();
4004 4008     input_size = fread(data,sizeof(PrimNum),MAX_INPUT_SIZE,stdin);
4005 4009
4010 +   PrimNum *basic_blocks[MAX_INPUT_SIZE];
4011 +   int numBBs = 0;
4012 +
4006 4013     PrimNum *start = data;
4007 4014     PrimNum *end   = data + input_size;
4008 4015
4009 4016     for ( PrimNum *pn = data; pn != end; pn++ ) {
4010 4017         if ( *pn == -1 ) {
4018 +       basic_blocks[numBBs] = pn;
4019 +       assert( data[pn-data] == -1 );
4020 +       numBBs++;
4021 +       }
4022 +   }
4023 +
4024 +   PrimNum **start2 = basic_blocks;
4025 +   PrimNum **end2   = basic_blocks + numBBs;
4026 +
4027 +   for ( ; start2 != end2; start2++ ) {
4028 +       PrimNum *pn = *start2;
4011 4029     optimize_rewrite( start, pn - start );
4012 4030     start = pn + 1;
```

Result

Inline cost codesize

```
...  ... @@ -3865,7 +3865,7 @@ void transitions(struct waypoint inst[], struct waypoint trans[])
3865 3865     struct waypoint *wo=&(inst[c->state_out]);
3866 3866     if (wo->cost == INF_COST)
3867 3867         continue;
3868 3868     - jcost = wo->cost + cost_codesize(s);
3868 3868     + jcost = wo->cost + priminfos[s].length;
3869 3869     if (jcost <= wi->cost) {
3870 3870         wi->cost = jcost;
3871 3871         wi->inst = s;
...  ... @@ -3937,7 +3937,7 @@ void optimize_rewrite(PrimNum origs[], int ninsts)
3937 3937
3938 3938     if (wo->cost == INF_COST)
3939 3939         continue;
3940 3940     - jcost = wo->cost + cost_codesize(s);
3940 3940     + jcost = wo->cost + priminfos[s].length;
3941 3941     if (jcost <= wi->cost) {
3942 3942         wi->cost = jcost;
3943 3943         wi->inst = s;
```

Result

Inlined generated hashfunction into lookup super.

Result

Global arrays to constants

```
... @@ -26,7 +26,7 @@ struct TableEntry {
26 26
27 27 static struct SuperState *lookup_super(PrimNum *start, int length)
28 28 {
29 - static unsigned short asso_values[] =
29 + static const unsigned short asso_values[] =
30 30 {
31 31     5,    0, 463, 453, 443, 433, 423, 413, 403, 393,
32 32     383, 373, 363, 353, 35,   70, 343, 333, 323, 313,
... @@ -55,7 +55,7 @@ static struct SuperState *lookup_super(PrimNum *start,
55 55     16, 11,   6,   1, 508, 503, 498, 493, 488, 483,
56 56     478, 473, 468, 128, 63, 510, 80
57 57 };
58 - static struct TableEntry wordlist[] =
58 + static const struct TableEntry wordlist[] =
59 59 {
```

Result

Printinst buffer

```
3824 + static char buffer[MAX_PRIM_NAME_LEN * MAX_SUPER * 2];
3825 +
3826 + char* out = buffer;
3827 +
3828 + *out = states[c->state_in];
3829 + out++;
3830 + *out = '_';
3831 + out++;
3832 +
3833 + PrimNum*      prims = super2 + c->offset;
3834 + PrimNum* const end  = prims + c->length;
3835 +
3836 + do {
3837 +     struct Name const* name = prim_names + *prims;
3838 +     strcpy( out, name->name );
3839 +     out += name->len;
3840 +     *out = '_';
3841 +     out++;
3842 +
3843 +     prims++;
3844 + } while ( prims != end );
3845 +
3846 + *out = states[c->state_out];
3847 + out++;
3848 + *out = ' ';
3849 + out++;
3850 + *out = 0;
3851 +
3852 + fputs( buffer, stdout );
```


Result

Printinst basic block into buffer

```
3854 3851 static inline void printBasicBlock() {  
3852 + *out = '\n';  
3853 + out++;  
3854 + *out = 0;  
3855 3855  
3856 + fputs( buffer, stdout );  
3857 + out = buffer;  
3856 3858 }  
3857 3859
```

Result

Printinst fputs replaced with write

```
3821 #define MAX_INPUT_SIZE 100000
3822 +
3823 +static char buffer[MAX_PRIM_NAME_LEN * MAX_SUPER * MAX_INPUT_SIZE * 2];
3822 3824 static char *out = buffer;
3823 3825
3824 3826 static inline void printinst(struct cost *c)
...    @@ -3851,9 +3853,10 @@ static inline void printinst(struct cost *c)
3851 3853 static inline void printBasicBlock() {
3852 3854     *out = '\n';
3853 3855     out++;
3854 - *out = 0;
3856 +// *out = 0;
3855 3857
3856 - fputs( buffer, stdout );
3858 + write( 1, buffer, out - buffer );
3859 +// fputs( buffer, stdout );
3857 3860     out = buffer;
3858 3861 ;
```

Printinst fputs replaced with write

Result

Printinst fputs replaced with fwrite

```
...    ... @@ -3853,9 +3853,9 @@ static inline void printinst(struct cost *c)
3853 3853 static inline void printBasicBlock() {
3854 3854     *out = '\n';
3855 3855     out++;
3856 3856 - *out = 0;
3856 3856 +// *out = 0;
3857 3857
3858 3858 - fputs( buffer, stdout );
3858 3858 + fwrite( buffer, 1, out - buffer, stdout );
3859 3859     out = buffer;
3860 3860 }
3861 3861
```

Printinst fputs replaced with fwrite

Result

Whole output into buffer

```
...    ... @@ -3853,10 +3853,6 @@ static inline void printinst(struct cost *c)
3853 3853 static inline void printBasicBlock() {
3854 3854     *out = '\n';
3855 3855     out++;
3856      -//  *out = 0;
3857      -
3858      - fwrite( buffer, 1, out - buffer, stdout );
3859      - out = buffer;
3860 3856 }
3861 3857
3862 3858 /* use dynamic programming to find the shortest paths within the basic
...    ... @@ -3979,6 +3975,12 @@ int main(int argc, char **argv, char **env)
3979 3975     start = pn + 1;
3980 3976     }
3981 3977     }
3978 +
3979 +//  *out = 0;
3980 +
3981 + fwrite( buffer, 1, out - buffer, stdout );
3982 +//  out = buffer;
3983 +
```


Result

Result