
Subject: Re: String::Cat optimization

Posted by [mirek](#) on Thu, 01 Dec 2011 19:41:26 GMT

[View Forum Message](#) <> [Reply to Message](#)

Tom1 wrote on Thu, 01 December 2011 08:50Hi,

I played around with Mirek's idea awhile and according to my simple '::GetTickCount()' benchmarking on MSC9/Win7x64 I managed to squeeze yet more performance out of it. The test covered all transfer lengths from 1 to 16 bytes.

The `svo_memcpy()` suffers a performance penalty at `len==16`, where secondary function call to `memcpy` steps in. The following macro approach helps dramatically to reduce that penalty. I also discovered that the `memcpy()` performance might not be reached systematically at transfer lengths above 11 bytes, so limiting the switch to `<= 11` bytes should improve overall performance.

```
inline void memcpy11i(char *t, const char *s, int len){
    switch(len) {
        case 11: t[10] = s[10];
        case 10: t[9] = s[9];
        case 9: t[8] = s[8];
        case 8: t[7] = s[7];
        case 7: t[6] = s[6];
        case 6: t[5] = s[5];
        case 5: t[4] = s[4];
        case 4: t[3] = s[3];
        case 3: t[2] = s[2];
        case 2: t[1] = s[1];
        case 1: t[0] = s[0];
    }
}
```

```
#define memcpy11(t, s, len) (len)>11 ? memcpy(t, s, len) : memcpy11i(t, s, len)
```

How does this perform on your systems?

Best regards,

Tom

Thanks, this is even better. Somehow I forgot that if compiler decides not to inline something, I can still force it by macro.

So, I am following your advice, using macro and limit:

```

#define SVO_MEMCPY(tgt, src, len) \
do { \
    const char *s__ = (const char *)(src); \
    char *t__ = (char *)(tgt); \
    switch(len) { \
    case 11: t__[10] = s__[10]; \
    case 10: t__[9] = s__[9]; \
    case 9: t__[8] = s__[8]; \
    case 8: t__[7] = s__[7]; \
    case 7: t__[6] = s__[6]; \
    case 6: t__[5] = s__[5]; \
    case 5: t__[4] = s__[4]; \
    case 4: t__[3] = s__[3]; \
    case 3: t__[2] = s__[2]; \
    case 2: t__[1] = s__[1]; \
    case 1: t__[0] = s__[0]; \
        break; \
    default: \
        memcpy(t__, s__, len); \
    } \
} while(false)

```

memcpy

TIMING Cat 40 : 1.98 s - 198.34 ns (2.15 s / 10000000), min: 0.00 ns, max: 1.00 ms,
 nesting: 1 - 10000000
 TIMING Cat 18 : 599.44 ms - 59.94 ns (767.00 ms / 10000000), min: 0.00 ns, max: 1.00
 ms, nesting: 1 - 10000000

inline svo_memcpy

TIMING Cat 40 : 1.45 s - 145.38 ns (1.63 s / 10000000), min: 0.00 ns, max: 1.00 ms,
 nesting: 1 - 10000000
 TIMING Cat 18 : 491.79 ms - 49.18 ns (671.00 ms / 10000000), min: 0.00 ns, max: 1.00
 ms, nesting: 1 - 10000000

macro

TIMING Cat 40 : 1.04 s - 103.71 ns (1.22 s / 10000000), min: 0.00 ns, max: 1.00 ms,
 nesting: 1 - 10000000
 TIMING Cat 18 : 302.09 ms - 30.21 ns (486.00 ms / 10000000), min: 0.00 ns, max: 1.00
 ms, nesting: 1 - 10000000

Amazing

It is interesting how year after year, we can still squeeze a bit from String....

Mirek
