XS Accessors Under The Hood

Sergey Aleynikov

YAPC Europe 2016

```perl
sub make_accessor {
    my ($class, $field) = @_;

    *{$class."::".$field} = sub {
        my $self = shift;

        if (scalar @_) {
            return $self->{$field} = $_[0];
        } else {
            return $self->{$field};
        }
    };
}
```

```perl
sub make_accessor {
    my ($class, $field) = @_;

    *{$class."::".$field} = sub {
        my $self = shift;

        if (scalar @_) {
            return $self->{$field} = $_[0];
        } else {
            return $self->{$field};
        }
    };
}
```

basic: Rounded run time per iteration: 3.1598e-07 $\sim$ 3200000/sec

hash: Rounded run time per iteration: 2.4837e-08 $\sim$ 40000000/sec

```perl
sub make_accessor {
    my ($class, $field) = @_;

    *{$class."::".$field} = sub {
        @_ > 1 ? ($_[0]->{$field} = $_[1]) : $_[0]->{$field}
    };
}
```

```perl
sub make_accessor {
    my ($class, $field) = @_;

    *{$class."::".$field} = sub {
        @_ > 1 ? ($_[0]->{$field} = $_[1]) : $_[0]->{$field}
    };
}
```

basic: Rounded run time per iteration: 3.1598e-07 ∼ 3200000/sec

better: Rounded run time per iteration: 2.6846e-07 ∼ 3700000/sec

```perl
sub make_accessor {
    my ($class, $field) = @_;

    *{$class."::".$field} = sub {
        @_ > 1 ? ($_[0]->{$field} = $_[1]) : shift->{$field}
    };
}
```

```perl
sub make_accessor {
    my ($class, $field) = @_;

    *{$class."::".$field} = sub {
        @_ > 1 ? ($_[0]->{$field} = $_[1]) : shift->{$field}
    };
}
```

better: Rounded run time per iteration: 2.6846e-07 ∼ 3700000/sec
shift: Rounded run time per iteration: 2.4664e-07 ∼ 4000000/sec
eval: Rounded run time per iteration: 2.2583e-07 ∼ 4400000/sec

```
perl -MO=Concise,-exec,foo \
 -e 'sub foo{ shift->{foo} }'

1  <;> nextstate(main 1 -e:1) v
2  <0> shift s*

3  <1> rv2hv sKR/1
4  <$> const(PV "foo") s/BARE
5  <2> helem sK/2
6  <1> leavesub[1 ref] K/REFC,1
```

```
perl -MO=Concise,-exec,foo \
 -e 'sub foo{ $_[0]->{foo} }'

1  <;> nextstate(main 1 -e:1) v
2  <$> gv(*_) s
3  <1> rv2av sKR/1
4  <$> const(IV 0) s
5  <2> aelem sKM/DREFHV,2
6  <1> rv2hv sKR/1
7  <$> const(PV "foo") s/BARE
8  <2> helem sK/2
9  <1> leavesub[1 ref] K/REFC,1
```

```
SV* name_sv = ST(0);
SV* key_sv = ST(1);

char* name = SvPV_nolen(name_sv);
CV* cv = newXS_flags(name, accessor, __FILE__, NULL, 0);
```

```
SV* name_sv = ST(0);
SV* key_sv = ST(1);

char* name = SvPV_nolen(name_sv);
CV* cv = newXS_flags(name, accessor, __FILE__, NULL, 0);

#define _XPVCV_COMMON
    union {
    OP *    xcv_start;
    ANY xcv_xsubany;
    }       xcv_start_u;

SvREFCNT_inc_simple_void_NN(key_sv);
CvXSUBANY(cv).any_ptr = key_sv;
```

```c
static MGVTBL sv_payload_marker;

char* name = SvPV_nolen(name_sv);
CV* cv = newXS_flags(name, accessor, __FILE__, NULL, 0);

sv_magicext(cv, key_sv,
    PERL_MAGIC_ext, &sv_payload_marker, NULL, 0);
SvRMAGICAL_off(cv);

CvXSUBANY(cv).any_ptr = key_sv;
```

```
char* name = SvPV_nolen(name_sv);
CV* cv = newXS_flags(name, accessor, __FILE__, NULL, 0);

STRLEN len;
char* key_buf = SvPV(key_sv, len);
SV* s_key_sv = newSVpvn_share(key_buf, len, 0);

sv_magicext(cv, s_key_sv,
    PERL_MAGIC_ext, &sv_payload_marker, NULL, 0);
SvREFCNT_dec_NN(s_key_sv);
SvRMAGICAL_off(cv);

CvXSUBANY(cv).any_ptr = s_key_sv;
```

```
XSPROTO(getter) {
    dXSARGS;
    SP -= items;

    SV* self;
    if (items != 1 || !SvROK(self = *++SP)) croak("Oops");

    SV* obj = SvRV(self);
    if (SvTYPE(obj) != SVt_PVHV) croak("Oops");

    SV* key_sv = CvXSUBANY(cv).any_ptr;
    HE* hent = hv_fetch_ent(obj, key_sv, 0, 0);

    if (hent) *SP = HeVAL(hent);
    else *SP = &PL_sv_undef;

    return;
}
```

```
XSPROTO(getter) {
    dXSARGS;
    SP -= items;

    SV* self;
    if (items != 1 || !SvROK(self = *++SP)) croak("Oops");

    SV* obj = SvRV(self);
    if (SvTYPE(obj) != SVt_PVHV) croak("Oops");

    SV* key_sv = CvXSUBANY(cv).any_ptr;
    HE* hent = hv_fetch_ent(obj, key_sv, 0, 0);

    if (hent) *SP = HeVAL(hent);
    else *SP = &PL_sv_undef;

    return;
}
```

eval: Rounded run time per iteration: 2.2583e-07 ∼ 4400000/sec
xs: Rounded run time per iteration: 8.454e-08 ∼ 12000000/sec
hash: Rounded run time per iteration: 2.4837e-08 ∼ 40000000/sec

```
perl -MO=Concise,-exec \
 -e '$foo->bar'
1  <0> enter
2  <;> nextstate v:{
3  <0> pushmark s
4  <$> gvsv(*foo) s

5  <.> method_named(PV "bar")
6  <1> entersub[t1] vKS/TARG
7  <@> leave[1 ref] vKP/REFC
```

```
perl -MO=Concise,-exec \
 -e '$foo->$bar'
1  <0> enter
2  <;> nextstate v:{
3  <0> pushmark s
4  <$> gvsv(*foo) s
5  <$> gvsv(*bar) s
6  <.> method K/1
7  <1> entersub[t1] vKS/TARG
8  <@> leave[1 ref] vKP/REFC
```

```
XSPROTO(getter) {
    OP* op = PL_op;

    if (
        (op->op_spare & 1) != 1 &&
        op->op_type == OP_ENTERSUB &&
        op->op_ppaddr == PL_ppaddr[OP_ENTERSUB]
    ) {
        op->op_spare |= 1;
        op->op_ppaddr = &getter_entersub; // :( Devel::NYTProf
    }

    ...
}
```

```
OP* getter_entersub(pTHX) {
    dSP;

    CV* cv = TOPs;
    if (cv != NULL && CvXSUB(cv) == &getter) {
        POPs; PUTBACK;
        getter(aTHX_ cv);
        return PL_op->op_next;
    }

    PL_op->op_ppaddr = PL_ppaddr[OP_ENTERSUB];
    return PL_ppaddr[OP_ENTERSUB](aTHX);
}
```

```
OP* getter_entersub(pTHX) {
    dSP;

    CV* cv = TOPs;
    if (cv != NULL && CvXSUB(cv) == &getter) {
        POPs; PUTBACK;
        getter(aTHX_ cv);
        return PL_op->op_next;
    }

    PL_op->op_ppaddr = PL_ppaddr[OP_ENTERSUB];
    return PL_ppaddr[OP_ENTERSUB](aTHX);
}
```

xs: Rounded run time per iteration: 8.454e-08 $\sim$ 12000000/sec

xse: Rounded run time per iteration: 6.1087e-08 $\sim$ 16500000/sec

```
perl -MO=Concise,-exec \
 -e '$foo->bar'
1  <0> enter
2  <;> nextstate v:{
3  <0> pushmark s
4  <$> gvsv(*foo) s

5  <.> method_named(PV "bar")
6  <1> entersub[t1] vKS/TARG
7  <@> leave[1 ref] vKP/REFC
```

```
perl -MO=Concise,-exec \
 -e '$foo->$bar'
1  <0> enter
2  <;> nextstate v:{
3  <0> pushmark s
4  <$> gvsv(*foo) s
5  <$> gvsv(*bar) s
6  <.> method K/1
7  <1> entersub[t1] vKS/TARG
8  <@> leave[1 ref] vKP/REFC
```

```
perl -MO=Concise,-exec \
 -e '$foo->bar'
1  <0> enter
2  <;> nextstate v:{
3  <0> pushmark s
4  <$> gvsv(*foo) s
5  <.> method_named(PV "bar")
6  <1> entersub[t1] vKS/TARG
7  <@> leave[1 ref] vKP/REFC
```

```
perl -MO=Concise \
 -e '$foo->bar'
7 <@> leave[1 ref] vKP/REFC ->(end)
1   <0> enter ->2
2   <;> nextstate ->3
6   <1> entersub[t1] vKS/TARG ->7
3     <0> pushmark s ->4
-     <1> ex-rv2sv sKM/1 ->5
4       <$> gvsv(*foo) s ->5
5     <.> method_named(PV "bar") ->6
```

```
XSPROTO(getter) {
    ...

    OP* methop = cUNOPx(op)->op_first;
    while (OpHAS_SIBLING(methop)) { methop = OpSIBLING(methop); }

    if (
        methop->op_next == op &&
        methop->op_type == OP_METHOD_NAMED &&
        methop->op_ppaddr == PL_ppaddr[OP_METHOD_NAMED]
    ) {
        methop->op_ppaddr = &getter_method_named;
    }

    ...
}
```

```
XSPROTO(getter) {
    ...

    OP* methop = cUNOPx(op)->op_first;
    while (OpHAS_SIBLING(methop)) { methop = OpSIBLING(methop); }

    if (
        methop->op_next == op &&
        methop->op_type == OP_METHOD_NAMED &&
        methop->op_ppaddr == PL_ppaddr[OP_METHOD_NAMED]
    ) {
        methop->op_ppaddr = &getter_method_named;
    }

    ...
}
```

eval: Rounded run time per iteration: 2.2583e-07 ~ 4400000/sec
xs: Rounded run time per iteration: 8.454e-08 ~ 12000000/sec
xse: Rounded run time per iteration: 6.1087e-08 ~ 16500000/sec
xsm: Rounded run time per iteration: 5.3485e-08 ~ 18700000/sec

Profile of bin/hailo for 1469s (of 1469s), executing 205334348 subroutine calls in 213 source files and 68 string evals.

/home/s.aleinikov/hailo/bin/hailo

Top 15 Subroutines

Calls	P	F	Exclusive Time	Inclusive Time	Subroutine
491407	1	1	208s	208s	DBI::st::fetch (xsub)
736471	2	2	204s	204s	DBI::st::execute (xsub)
1472932	6	1	42.2s	60.0s	DBIx::Class::ResultSet::_resolved_attrs
491409	1	1	34.8s	50.8s	DBIx::Class::Storage::DBIHacks::_resolve_column_info
4418892	12	12	31.0s	43.2s	Class::Accessor::Grouped::get_inherited
736465	1	1	29.5s	75.2s	DBIx::Class::ResultSet::search_rs
2455768	9	7	28.2s	339s	Try::Tiny::try
1472928	1	1	24.7s	68.6s	SQL::Abstract::_where_hashpair_SCALAR
1472930	2	2	23.0s	68.9s	DBIx::Class::ResultSet::new
736465	1	1	21.3s	34.4s	DBIx::Class::Storage::DBI::SQLite::_dbi_attrs_for_bind
3437268	7	2	19.2s	78.1s	DBIx::Class::SQLMaker::_quote
736465	1	1	18.6s	114s	SQL::Abstract::_where_HASHREF
5646690	9	6	18.6s	28.6s	next::method
491407	1	1	17.3s	732s	DBIx::Class::Storage::DBI::Cursor::next
3437270	3	1	17.3s	26.8s	SQL::Abstract::_assert_pass_injection_guard

See all 4412 subroutines

Top 15 Subroutines

Calls	P	F	Exclusive Time	Inclusive Time	
214873	3	3	1.67s	2.31s	Class::Accessor::Grouped::get_inherited
22124	2	2	1.16s	1.29s	
1132	2	1	925ms	925ms	
17578	9	6	756ms	14.6s	
47	1	1	703ms	703ms	
132407	211	24	688ms	8.76s	
132407	1	1	681ms	8.03s	
354	1	1	681ms	848ms	
17105	16	7	651ms	1.74s	DBIx::Class::ResultSet::search_rs
67758	1	1	615ms	5.46s	DBIx::Class::InflateColumn::get_inflated_column
88709	2	1	512ms	1.54s	DBIx::Class::ResultSourceProxy::column_info
24996	35	17	495ms	1.80s	
4	1	1	481ms	13.9s	
57397	1	1	469ms	543ms	Class::Accessor::Grouped::get_super_paths
27162	3	2	443ms	1.40s	

See all 24547 subroutines

```
use CAIXS;

my $LOADED_CACHE = {};
sub _component_class_on_read {
    my $class = $_[0];

    if (defined $class && !blessed($class) && !exists $LOADED_CACHE->{$class}) {
        $LOADED_CACHE->{$class} = undef;
        DBIx::Class::Componentised->ensure_class_loaded($class);
    }

    return $class;
}

sub _inherited_ro_instance_on_write {
    $_[0]->throw_exception("Cannot set value on an instance") if blessed $_[0];
    return $_[1];
}

Class::Accessor::Inherited::XS::register_types(
    component_class       => {read_cb  => \&_component_class_on_read},
    inherited_ro_instance => {write_cb => \&_inherited_ro_instance_on_write},
);
```

```
*DBIx::Class::AccessorGroup::mk_group_accessors = sub {
    my ($self, $group, @fields) = @_;
    my $class = ref($self) ? ref($self) : $self;

    state $known_types = {
        inherited               => 'inherited',
        inherited_ro_instance   => 'inherited_ro_instance',
        component_class         => 'component_class',
        simple                  => 'object',
    };

    if (my $type = $known_types->{$group}) {
        Class::Accessor::Inherited::XS::mk_type_accessors($class, $type, @fields);

    } else {
        goto &Class::Accessor::Grouped::mk_group_accessors;
    }
};

*DBIx::Class::mk_classaccessor = sub {
    my ($self, $field) = @_;
    $self->mk_group_accessors('inherited', $field);

    $self->$field($_[2]) if scalar @_ > 2;
};
```

Questions?

https://github.com/dur-randir/