Benchmarking ADT struct representation vs Vector struct representation

This notebook compares the representation of structures on Boogie level as:

- Vectors of the universal `$Value` type. Values for all fields are represented in boxed ($Value) representation. Selecting and updating fields amounts to vector indexing. On select/update values need to be unboxed/boxed.
- Abstract data types. Values of fields are stored in unboxed representation unless their type is generic. Equality on universal values has to be implemented by a large case distinction of the multiple ADT variants. However, equality is extensional unless a struct contains a transient field of vector type, which breaks extensionality.

Preparation

Load the prover-lab crate. This may take *long* (minutes) the first time the Jupyter server is started because it compiles a lot Rust sources.

In [2]:
```
:sccache 1
:dep prover-lab = { path = "../.." }
```

Out[2]:
```
sccache: true
```

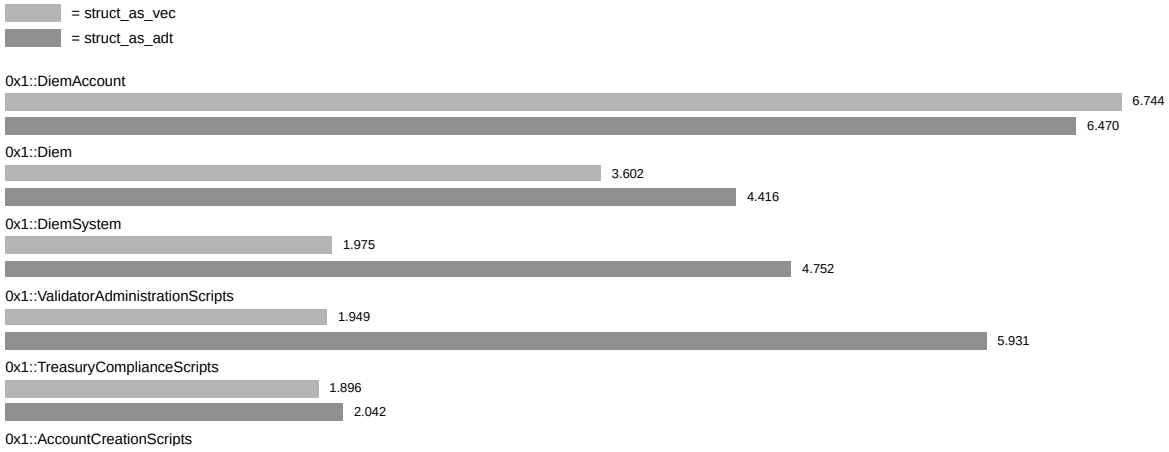
Make functions from the benchmark module available:

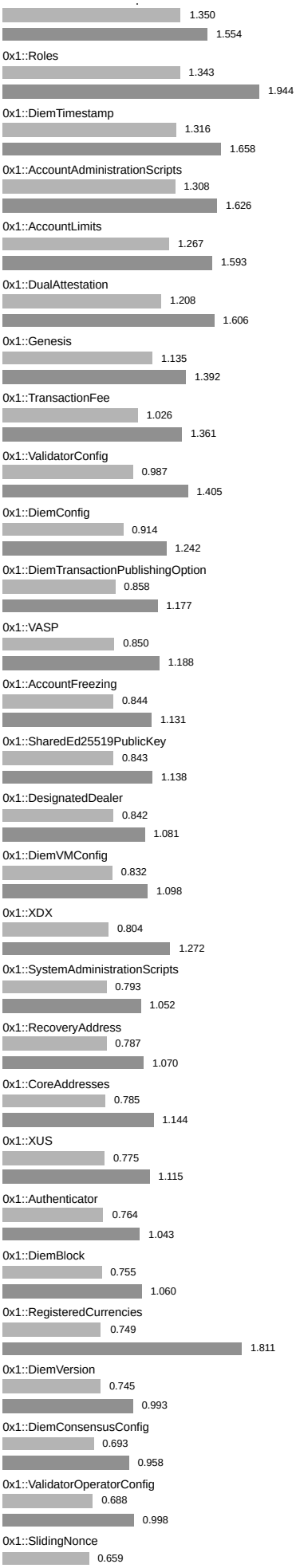
In [3]:
```
use prover_lab::benchmark::*;
```

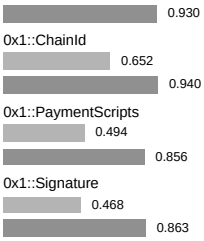
Module Verification Time

In [6]:
```
let mut struct_as_vec_mod = read_benchmark("struct_as_vec.mod_data")?;
let mut struct_as_adt_mod = read_benchmark("struct_as_adt.mod_data")?;
struct_as_vec_mod.sort(); // Will also determine order of other samples.
plot_benchmarks(&[&struct_as_vec_mod, &struct_as_adt_mod])
```

Out[6]:




1.350
1.554
0x1::Roles
1.343
1.944
0x1::DiemTimestamp
1.316
1.658
0x1::AccountAdministrationScripts
1.308
1.626
0x1::AccountLimits
1.267
1.593
0x1::DualAttestation
1.208
1.606
0x1::Genesis
1.135
1.392
0x1::TransactionFee
1.026
1.361
0x1::ValidatorConfig
0.987
1.405
0x1::DiemConfig
0.914
1.242
0x1::DiemTransactionPublishingOption
0.858
1.177
0x1::VASP
0.850
1.188
0x1::AccountFreezing
0.844
1.131
0x1::SharedEd25519PublicKey
0.843
1.138
0x1::DesignatedDealer
0.842
1.081
0x1::DiemVMConfig
0.832
1.098
0x1::XDX
0.804
1.272
0x1::SystemAdministrationScripts
0.793
1.052
0x1::RecoveryAddress
0.787
1.070
0x1::CoreAddresses
0.785
1.144
0x1::XUS
0.775
1.115
0x1::Authenticator
0.764
1.043
0x1::DiemBlock
0.755
1.060
0x1::RegisteredCurrencies
0.749
1.811
0x1::DiemVersion
0.745
0.993
0x1::DiemConsensusConfig
0.693
0.958
0x1::ValidatorOperatorConfig
0.688
0.998
0x1::SlidingNonce
0.659

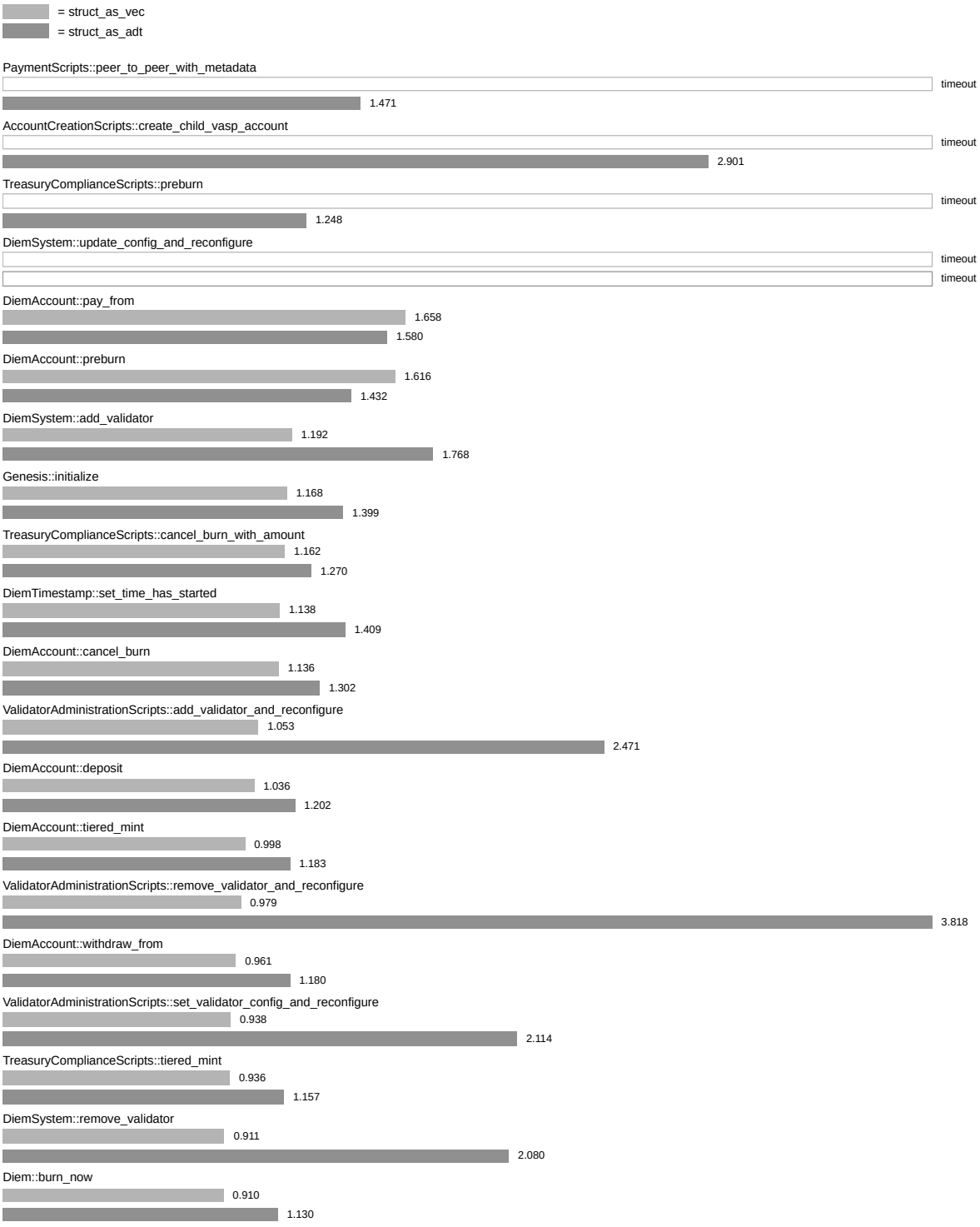


Top 20 by Function

In [7]:

```
let mut struct_as_vec_fun = read_benchmark("struct_as_vec.fun_data")?;
let mut struct_as_adt_fun = read_benchmark("struct_as_adt.fun_data")?;
struct_as_vec_fun.sort(); // Will also determine order of other samples.
struct_as_vec_fun.take(20);
plot_benchmarks(&[&struct_as_vec_fun, &struct_as_adt_fun])
```

Out[7]:


= struct_as_vec
= struct_as_adt
PaymentScripts::peer_to_peer_with_metadata
timeout
1.471
AccountCreationScripts::create_child_vasp_account
timeout
2.901
TreasuryComplianceScripts::preburn
timeout
1.248
DiemSystem::update_config_and_reconfigure
timeout
timeout
DiemAccount::pay_from
1.658
1.580
DiemAccount::preburn
1.616
1.432
DiemSystem::add_validator
1.192
1.768
Genesis::initialize
1.168
1.399
TreasuryComplianceScripts::cancel_burn_with_amount
1.162
1.270
DiemTimestamp::set_time_has_started
1.138
1.409
DiemAccount::cancel_burn
1.136
1.302
ValidatorAdministrationScripts::add_validator_and_reconfigure
1.053
2.471
DiemAccount::deposit
1.036
1.202
DiemAccount::tiered_mint
0.998
1.183
ValidatorAdministrationScripts::remove_validator_and_reconfigure
0.979
3.818
DiemAccount::withdraw_from
0.961
1.180
ValidatorAdministrationScripts::set_validator_config_and_reconfigure
0.938
2.114
TreasuryComplianceScripts::tiered_mint
0.936
1.157
DiemSystem::remove_validator
0.911
2.080
Diem::burn_now
0.910
1.130

In [ ]: