

Pack v4

# Pack v4 Goals

Faster runtime execution

# Pack v4 Goals

Faster runtime execution

i.e. make this not suck:

```
git rev-list HEAD -- include/asm-m68k
```

# Pack v4 Goals

Faster runtime execution

i.e. make this not suck:

```
git rev-list HEAD --include/asm-m68k
```

Improved compression  
(smaller size) is not a goal

# Pack v4 Goals

Faster runtime execution

i.e. make this not suck:

```
git rev-list HEAD -- include/asm-m68k
```

Improved compression  
(smaller size) is not a goal  
But of course smaller size == less  
page faults == faster

# Current Bottlenecks

Last profiling showed:

`inflate()` = ~30%

`get_sha1_hex()` = ~15-30% (I forget how much exactly)

# High Level Changes

Reformat commits and trees when in pack format

# High Level Changes

Reformat commits and trees when in pack format  
(loose format unmodified)

# High Level Changes

Reformat commits and trees when in pack format  
(loose format unmodified)

Trees:

- fixed length record: (name\_index, sha1)

# High Level Changes

Reformat commits and trees when in pack format  
(loose format unmodified)

Trees:

- fixed length record: (name\_index, sha1)
- name\_index: 2 byte offset into a name dictionary
- sha1: 20 byte SHA-1

# High Level Changes

Reformat commits and trees when in pack format  
(loose format unmodified)

Trees:

- fixed length record: (name\_index, sha1)
- name\_index: 2 byte offset into a name dictionary
- sha1: 20 byte SHA-1

Name Dictionary:

- Deflated string table

# High Level Changes

Reformat commits and trees when in pack format  
(loose format unmodified)

Trees:

- fixed length record: (name\_index, sha1)
- name\_index: 2 byte offset into a name dictionary
- sha1: 20 byte SHA-1

Name Dictionary:

- Deflated string table
- Entries are "classical": mode <SP> name <NUL>

# Classic Tree Parsing

```
if (parse_tree(tree) < 0)
    die("bad tree %s", sha1_to_hex(obj->sha1));

init_tree_desc(&desc, tree->buffer, tree->size);
while (tree_entry(&desc, &entry)) {
    switch (object_type(entry.mode)) {
```

.....

# Classic Tree Parsing

```
if (parse_tree(tree) < 0)
    die("bad tree %s", sha1_to_hex(obj->sha1));

init_tree_desc(&desc, tree->buffer, tree->size);
while (tree_entry(&desc, &entry)) {
    switch (object_type(entry.mode)) {
```

.....

Buffer assumed to be in canonical form (current tree format)

# Classic Tree Parsing

```
if (parse_tree(tree) < 0)
    die("bad tree %s", sha1_to_hex(obj->sha1));

init_tree_desc(&desc, tree->buffer, tree->size);
while (tree_entry(&desc, &entry)) {
    switch (object_type(entry.mode)) {
```

.....

Buffer assumed to be in canonical form (current tree format)  
Needs to recognize pack v4 tree format for improved efficiency

# Classic Tree Parsing

```
if (parse_tree(tree) < 0)
    die("bad tree %s", sha1_to_hex(obj->sha1));

init_tree_desc(&desc, tree->buffer, tree->size);
while (tree_entry(&desc, &entry)) {
    switch (object_type(entry.mode)) {
```

.....

Buffer assumed to be in canonical form (current tree format)  
Needs to recognize pack v4 tree format for improved efficiency

Nico and I want:

# Classic Tree Parsing

```
if (parse_tree(tree) < 0)
    die("bad tree %s", sha1_to_hex(obj->sha1));

init_tree_desc(&desc, tree->buffer, tree->size);
while (tree_entry(&desc, &entry)) {
    switch (object_type(entry.mode)) {
```

.....

Buffer assumed to be in canonical form (current tree format)  
Needs to recognize pack v4 tree format for improved efficiency

Nico and I want:

- to parse pack v4 trees directly when possible

# Classic Tree Parsing

```
if (parse_tree(tree) < 0)
    die("bad tree %s", sha1_to_hex(obj->sha1));

init_tree_desc(&desc, tree->buffer, tree->size);
while (tree_entry(&desc, &entry)) {
    switch (object_type(entry.mode)) {
```

.....

Buffer assumed to be in canonical form (current tree format)  
Needs to recognize pack v4 tree format for improved efficiency

Nico and I want:

- to parse pack v4 trees directly when possible
- to walk the tree delta chain during parsing, not ahead of time

# Pack v4 Roadblocks

Need to refactor `tree_entry()`

# Pack v4 Roadblocks

Need to refactor `tree_entry()`

`sha1_file.c` must be able to export the encoded tree directly

# Pack v4 Roadblocks

Need to refactor `tree_entry()`

`sha1_file.c` must be able to export the encoded tree directly

`tree_entry()` has to consume either tree

# Pack v4 Roadblocks

Need to refactor `tree_entry()`

`sha1_file.c` must be able to export the encoded tree directly

`tree_entry()` has to consume either tree

Callers of `tree_entry()` can't use `read_sha1_file()`