Reference-count design for elements of lists/arrays protected by RCU.

Reference counting on elements of lists which are protected by traditional reader/writer spinlocks or semaphores are straightforward:

1.
```
add()
{
    alloc_object
    ...
    atomic_set(&el->rc, 1);
    write_lock(&list_lock);
    add_element
    ...
    write_unlock(&list_lock);
}
```

2.
```
search_and_reference()
{
    read_lock(&list_lock);
    search_for_element
    atomic_inc(&el->rc);
     ...
    read_unlock(&list_lock);
    ...
}
```

3.
```
release_referenced()
{
    ...
    atomic_dec(&el->rc, relfunc)
    ...
}
```

4.
```
delete()
{
    write_lock(&list_lock);
    ...
    delete_element
    write_unlock(&list_lock);
    ...
    if (atomic_dec_and_test(&el->rc))
        kfree(el);
    ...
}
```

If this list/array is made lock free using RCU as in changing the write_lock() in add() and delete() to spin_lock() and changing read_lock() in search_and_reference() to rcu_read_lock(), the atomic_inc() in search_and_reference() could potentially hold reference to an element which has already been deleted from the list/array. Use atomic_inc_not_zero() in this scenario as follows:

1.
```
add()
{
    alloc_object
    ...
    atomic_set(&el->rc, 1);
    spin_lock(&list_lock);

    add_element
    ...
    spin_unlock(&list_lock);
}
```

2.
```
search_and_reference()
{
    rcu_read_lock();
    search_for_element
    if (!atomic_inc_not_zero(&el->rc)) {
        rcu_read_unlock();
        return FAIL;
    }
    ...
    rcu_read_unlock();
}
```

3.
```
release_referenced()
{
    ...
    if (atomic_dec_and_test(&el->rc))
        call_rcu(&el->head, el_free);
    ...
}
```

4.
```
delete()
{
    spin_lock(&list_lock);
    ...
    delete_element
    spin_unlock(&list_lock);
```

```
}
                                        ...
                                        if (atomic_dec_and_test(&el->rc))
                                            call_rcu(&el->head, el_free);
                                        ...
                                    }
```

Sometimes, a reference to the element needs to be obtained in the update (write) stream. In such cases, atomic_inc_not_zero() might be overkill, since we hold the update-side spinlock. One might instead use atomic_inc() in such cases.