Item 225
git_comments:

git_commits:

1. **summary:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   message: KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov… …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging Author: Grant Henke <granthenke@gmail.com> Author: Grant Henke <granthenke@users.noreply.github.com> Author: Ismael Juma <ismael@juma.me.uk> Reviewers: Flavio Junqueira, Jun Rao, Ismael Juma, Gwen Shapira Closes #1006 from granthenke/simple-authorizer-fix

github_issues:

github_issues_comments:

github_pulls:

1. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
2. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
   label: code-design
3. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
4. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
5. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
6. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
7. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
8. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
9. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
   body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
10. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache

when modifying ACLs - Add debug logging
11. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
12. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
13. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
14. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
15. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
16. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
17. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
18. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
19. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
20. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
21. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
22. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
23. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
24. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging

25. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
26. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
27. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
28. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
29. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
30. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
31. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: test
32. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
33. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
34. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
35. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
36. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
37. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
38. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging

39. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
40. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
41. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
42. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
43. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
44. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
45. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
46. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
47. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
48. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
49. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
50. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
51. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
52. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache

when modifying ACLs - Add debug logging
label: code-design

53. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
54. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
55. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
56. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
57. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
58. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
59. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
60. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
61. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
62. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
63. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
64. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
65. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
66. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging

label: documentation

67. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
68. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
69. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
70. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
71. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
72. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
73. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
74. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
75. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: documentation
76. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
77. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
78. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
    label: code-design
79. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
    body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging

label: code-design

80. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
81. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
82. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
83. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
84. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
85. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
86. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
87. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
88. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
89. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
90. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
91. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
92. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
93. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache

when modifying ACLs - Add debug logging
label: code-design

94. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
95. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
96. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
97. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
98. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
99. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
100. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
101. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
102. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
label: code-design
103. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
104. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
105. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
106. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging

107. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
     body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
     label: code-design
108. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
     body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
     label: code-design
109. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
     body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
110. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
     body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging
111. **title:** KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov…
     body: …e calls Changes the SimpleAclAuthorizer to: - Track and utilize the zookeeper version when updating zookeeper to prevent data loss in the case of stale reads and race conditions - Update local cache when modifying ACLs - Add debug logging

github_pulls_comments:

1. This change will now make a call to zookeeper for every add/remove call. I am assuming that the volume of ACL modification requests to the authorizer would be quite low, therefore this is okay. If thats not the case, let me know. I think this issue can still occur with multiple Zookeeper nodes in the case where the client gets a stale zookeeper read. I am not sure the best way to handle this with Zookeeper and our packaged client. @fpj do you have any recommendations on making sure the "read, update, write" pattern is synchronized/safe? Do we need a completely different approach? (links or research material are welcome)
2. **body:** @ijuma @SinghAsDev @fpj I updated the patch to be "safe". So that stale reads and race conditions are handled. It does this by tracking its expected zookeeper version and handling write failures. Its also more optimistic about its cached values instead of reading from zookeeper every time. I also added a test that concurrently updates the same node with 110 mixed requests. This definitely makes the SimpleAclAuthorizer more safe, but perhaps less "simple".
   label: code-design
3. @ijuma @Parth-Brahmbhatt @SinghAsDev @fpj @ewencp @gwenshap Could any of you review this today?
4. @granthenke I'm not sure I'll have time to get to it today, but I can review it by wed. To answer your question above, it doesn't matter if the read is stale if you're using a conditional set, then the write won't go through if the version isn't the one expected. In the worst case, the broker will have to try it again. One thing you can goto reduce the probability of a stale read is to issue a sync request right before the read. This way you'll be forcing all pending updates from the leader to the follower to go through. You can't completely avoid the issue though because you can still have concurrent writes even when syncing.
5. @fpj Thanks for the follow up and confirmation. Using the versioned requests is the approach I took in the "rewrite" after my initial question where I mentioned you. I had considered sending a sync request, and would still like to add that, but unfortunately the ZkClient we use does not support it. It also doesn't support a versioned delete request, which is unfortunate. Perhaps I should open a PR for that.
6. @granthenke I'll see if I can suggest something concrete, but in the past I've bypassed zkclient and used the zk handle directly, it might be doable in your case as well.
7. @fpj Thanks for the review. I updated the patch to use the versioned delete and handle the empty data edge cases.
8. @fpj Any further comments? @ewencp @gwenshap @junrao Could one of you review?
9. Sorry for the delay @granthenke, good catch spotting this serious problem. I did a review pass and left a few comments.
10. @granthenke I have created a PR for the conditional delete issue in zkclient in the case you want to track it or chime in: https://github.com/sgroschupf/zkclient/pull/46 I'll check with them how fast they will be

able to produce a new release.
11. @fpj Cool! I just started working on that too :). I will check it out and chime in.
12. @granthenke The conditional delete patch has been merged into zkclient. Do you want to give it a try with a build of zkclient master to see if it is good?
13. @fpj I tested locally on 0.8-dev and get a lot of test failures. The failures look like they are related to a change that went into zkclient after your change: https://github.com/sgroschupf/zkclient/commit/2c8f4e907b22b16455f781aa518a85f6ec0c18fc The stacktrace from the tests is: ``` org.I0Itec.zkclient.exception.ZkAuthFailedException: Authentication failure at org.I0Itec.zkclient.ZkClient.waitForKeeperState(ZkClient.java:946) at org.I0Itec.zkclient.ZkClient.waitUntilConnected(ZkClient.java:925) at org.I0Itec.zkclient.ZkClient.connect(ZkClient.java:1230) at org.I0Itec.zkclient.ZkClient.<init> (ZkClient.java:156) at org.I0Itec.zkclient.ZkClient.<init>(ZkClient.java:130) at kafka.utils.ZkUtils$.createZkClientAndConnection(ZkUtils.scala:80) at kafka.utils.ZkUtils$.apply(ZkUtils.scala:62) at kafka.zk.ZooKeeperTestHarness$class.setUp(ZooKeeperTestHarness.scala:67) at kafka.server.LogOffsetTest.setUp(LogOffsetTest.scala:48) java.lang.NullPointerException at kafka.server.LogOffsetTest.tearDown(LogOffsetTest.scala:59) ```
14. @granthenke I also made that change to throw an exception when auth fails. I need to have a closer look to see why it is throwing an exception in those cases.
15. @granthenke It is fixed in 0.8-dev, I checked it locally myself. Let me know if it works for you too. I was also thinking that we might need to switch to zkclient 0.8 first, before checking in this patch. Otherwise, we will break trunk.
16. **body:** @fpj The patch as currently written still uses the workaround. The change for zkclient 0.8 is still local. This patch fixes a big enough issue that I think it should get in for the next release with or without zkclient 0.8. If that means committing it as is, and then a follow up when 0.8 is available that works for me. But even with the un-optimal delete behavior, its still safer than it was before.
    label: code-design
17. **body:** @granthenke I was assuming that you want to have the conditional delete in with this patch, but if it is not the case, then sure, we don't need to switch to 0.8 with the workaround. I'm definitely not disputing the importance of the issue.
    label: code-design
18. @fpj I do want the conditional delete. I get that via `zkUtils.zkConnection.getZookeeper.delete(path, expectedVersion)` like you recommended. I think that is okay until zkclient 0.8 is available.
19. @granthenke Ok, just to clarify, the zkclient folks will release 0.8 as soon as we say that the changes work for us, so if you want to have this in for 0.10, we need to make sure that the changes work so that we can flag it to them. Once they release, we upgrade kafka to use zkclient 0.8 and assuming everyone is happy with this PR, we can merge it, in this order. Does it sound right to you?
20. @fpj Gotcha. I didn't understand that we had control over the zkclient release. I will test today and report back. Unless you think it should be 2 separate patches, I am happy to include the 0.8 bump in this patch.
21. @granthenke I was proposing two patches to separate concerns, but you're right that we could just do it all here.
22. @fpj I can do it in 2 steps. I will open a jira to track the upgrade change and make this depend on it.
23. @fpj I created KAFKA-3403 to track the upgrade. I tested zkclient locally and everything passes.
24. @granthenke ok, sounds good. do you think you can test a version of your patch with the conditional delete call? It'd be good to make sure that we don't have anything unexpected due to the conditional delete changes.
25. @fpj Yeah, that is what I tested.
26. @fpj I have merged with trunk and updated this patch to use the ZkClient versioned delete.
27. Tests pass locally. Jenkins failure looks unrelated. @fpj @ijuma @junrao Would you guys be able to give this a final review today?
28. @granthenke it looks great, thank you for updating. I left a few comments in the case you have a chance to go over them.
29. @fpj thanks for looking over it again. I responded to some of your requests.
30. @fpj I addressed your comments and updated the code.
31. **body:** @granthenke looks like the concurrency test fails because of too many attempts (one of the latest changes) : ``` java.lang.IllegalStateException: Failed to update ACLs for Topic:test after trying a maximum of 15 times) ``` I suppose that for any number you choose, there is a chance that the test case will need that number or more, which is a bit concerning because it can keep running for some time. I

think we will need to back off in the case of collision, like with a random sleep if you want something simple. Perhaps you have a better idea?
label: test

32. **body:** @fpj Causing collisions is the point of the test. It verifies that the code always maintains the correct state no matter how many retries it takes. I really don't want to over engineer this and I don't think this is a real world issue. The test just goes to extremes to prove ACLs are never lost. Do you feel strongly about the arbitrary retry limit code?
    label: code-design
33. **body:** @granthenke Even if this loops forever, this is just AclCommand running, right? In the very worst case of running for too long due to collisions, you'd have to kill it and try again. There is little harm in not having the retries if that's the case, unless it is problem that you're not sure whether the ACL change has gone through or not, but I suppose you can check if needed. I still think that it'd be better to back off in the case of collisions rather than running in a tight loop, but I don't want to block this issue on it. Perhaps we can create a jira and work on it later.
    label: code-design
34. @fpj I can add a backoff over the next few days. Its only quick in the test because we are on an EmbeddedZookeeper. In the real world the read of the data happens before the next update request which slows things down.
35. @fpj I added a simple backoff with a random jitter
36. LGTM. Test failure unrelated (and looks like a Jenkins issue) - validated tests on my machine. I love the new concurrency tests util - should be very handy in the future.
37. Thanks for the work on the patch @granthenke.
38. Thanks for all the review @fpj!

github_pulls_reviews:

1. Accidental?
2. **body:** Ah yeah, I do that sometimes to view various logging statements for testing. I removed the debug statements, so I should remove this.
    label: code-design
3. Maybe add a test for adding acl from one authorizer and removing from other and making sure it gets reflected.
4. Probably aclChangedFlag should be set in updateCache.
5. I can add something like that, but will need to think through it. I used 2 additions, because it clearly shows "whats missing". I can add the test for the add & delete but in the case where the add was "dropped" checking for no acl would still be true.
6. **body:** That would result in an infinite loop of change notifications. Since every change notification calls `updateCache`.
    label: code-design
7. Shouldn't we be updating cache in `AclChangedNotificationHandler.processNotification` only when changes are not made by us. So, in `updateAclChangedFlag` should update last update i (update_id), which can be path returned by `ZkUtils.createSequentialPersistentPath`, Now, `AclChangedNotificationHandler.processNotification` can ignore any updates that were made prior to update_id.
8. **body:** We could definitely do something like that as an optimization. There may be a better way to "batch-up" frequent notifications in general too, to reduce chattiness with Zookeeper. I may open a separate jira for optimizations, since thats not related to this bug fix. (unless we need to rewrite all this to fix the bug)
    label: code-design
9. Sure, that works!
10. Just curious, why -2?
11. **body:** I should clean, that up. I just needed some way to indicated there is no version yet, and -1 in the Zookeeper api means "ignore the version". I can rework the code to eliminate this, or use Option/None.
    label: code-design
12. I'm not sure why you want to spin it until the write is complete. If there is an error and the operation does not complete, then perhaps it should be up to the caller to call it again. Could you elaborate on the case you have in mind and that makes this while loop necessary?
13. **body:** I agree that you do want to use the version here. The options I see are that we get the handle directly through ZkUtils.zkConnection.getZookeeper and call delete using the raw zk handle or that we

patch ZkClient. I had a look at ZkClient and it should be straightforward to do it. Perhaps we can do the former until we have a new version of ZkClient with the versioned delete call and propose the change to ZkClient in parallel. I can help with that if you like the direction.
label: code-design

14. This goal of this while loop is to retry the update to zookeeper if we fail based on the zookeeper version. The zookeeper version could be incorrect if the cached version in this instance is not up to date. That is most likely to occur if there are concurrent updates on a separate instance of the authorizer. If the failure is based on anything other than version, or the handled exceptions. Then an exception will be thrown and propagated back to the user. Ideally I would use the sync call to prevent any "spinning" but the ZkClient we use does not have that method (https://github.com/sgroschupf/zkclient/issues/44).
15. There is a corner case here I think. If data is empty, then you don't want to create the znode. I don't think you're expecting such a call, which makes sense, but there isn't anything that prevents a caller from making that mistake.
16. Good catch I will fix that.
17. The problem with sync is that it is an asynchronous call and I believe ZkClient only supports the synchronous API. I don't think it is worth using the sync call here because it won't stop concurrent accesses anyway. When we move to the async API for the controller, then perhaps we can make this change here.
18. Yeah, that makes sense. Thanks for clarifying.
19. Another thing to watch out for here is that the delete can also throw a no node exception, and in this case you also don't want to create the znode. I suppose that there will be no data to write like in my previous comment, so it should be fine, but I wanted to raise the point.
20. **body:** I created an issue on the ZkClient to track that change (https://github.com/sgroschupf/zkclient/issues/45). In the mean time I will use the workaround and add a comment/jira to update it later.
    label: code-design
21. If newAcls is empty, then we'll do this update to later delete? Can we spot it earlier so that we do a single ZK operation?
22. Sounds good.
23. Yes we can, the existing implementation was based on the limitation of not having the versioned delete.
24. In the updated patch, I handle empty data and No node exceptions in the delete call.
25. Map also has a `getOrElse` so you can replace `get` followed by `getOrElse` with a single `getOrElse`.
26. Hmm, if we use the raw ZK api here directly, we will need to handle the ConnectionLossException which zkclient hides from us. Perhaps we should just patch zkclient to expose versioned deleted and have zkclient release a new version.
27. **body:** Could we add a comment on what version is?
    label: documentation
28. **body:** indentation
    label: code-design
29. Could we add space after if?
30. Is this necessary? If we successfully updated ZK, it's probably simpler to always propagate the change even if the content is identical.
31. Its not required, just an optimization. I am happy to remove it.
32. @junrao Upon further review this is required to support the delete contract. It returns a boolean that indicates if any acls were actually deleted. I can move the propagation outside of the check, but I will still need the check to return true or false. Should I move the propagation out of the acls change check?
33. **body:** I can change the variable to `zkVersion` to make it more clear.
    label: code-design
34. @junrao This workaround was suggested by @fpj as an intermediate solution until that change occurs. I filed an issue in the ZkClient project. (https://github.com/sgroschupf/zkclient/issues/45)
35. **body:** This should probably be in the `SimpleAclAuthorizer` object since it does not need access to the instance.
    label: code-design
36. **body:** Typo: `ensuring`.
    label: documentation
37. **body:** Out of curiosity, is there a reason you used `diff` instead of the more common `--`?
    label: code-design
38. I'm a bit confused by this statement. We return a boolean indicating if an update was made so it seems a bit contradictory?

39. **body:** Nitpick: space after `if`. It would be good to check the rest of the code for this.
    label: code-design
40. **body:** Nitpick: I think we should write it as `ACLs` when logging.
    label: code-design
41. It seems like we never use the value initially assigned to this, right? I'd make that explicit and assign null.
42. When a path is deleted, and then recreated, does the version start from 0 again? Is it possible that given a particular sequence of events, this could lead to lost updates? This has probably been considered already, but I thought I would ask to make sure I understand this correctly. cc @fpj @granthenke
43. It would be better to take functions instead of runnables. Also, maybe use timeout in ms since that's more common in Kafka.
44. Is it really worth doing all of this instead of just calling `ExecutorService.invokeAll`? Given thread scheduling which we don't control, I am not sure we gain much.
45. Not sure we should bother with `List` instead of `Seq` (calling code doesn't have to do the conversion then).
46. I think the code does this to allow for initialization before running. That way the actual "action" happens all at once and increases the chance of exposing concurrency issues. I didn't write this logic from scratch. I took it from the link in the comment. I figured this small piece of test code was better than adding a dependency.
47. @ijuma In this code setting that version is more of a formality. When we update the cache, the entry is removed. So no entry and no version is set.
48. **body:** Often I prefer the named method over the symbolic one. I find it can be more clear, especially to those less familiar with Scala. `diff` just calls `--`. Happy to change it if you think I should
    label: code-design
49. The boolean indicates if the ACLs content changed. Regardless we need to update zookeeper to check/increment the version at a minimum. I can clarify this doc.
50. **body:** I understand what it's trying to do. My point is that we have no control when things actually run, the kernel does. So we don't know when a thread will be rescheduled after the countdown latch is open. As such, I am not sure if this actually buys us much (if anything) compared to the standard way. One thing that this does probably is to randomise the order a little more, but we could do that trivially ourselves before submitting to the executor.
    label: code-design
51. **body:** I think `--` is clearer in this case (it's widely used as the opposite of `++`). `diff` is rarely used in comparison in my experience,
    label: code-design
52. @ijuma This code works and has some nice functionality around tracking timeouts and exceptions. I didn't write it from scratch, as indicated in the comment. I don't want to spend time optimizing this, unless you feel really strongly about it. If you like you can follow up with a patch that re-writes its internals.
53. It just seems that 30 lines for something that can be done in less than 10 is not optimal. The signature of the method I am referring to is: ``` java <T> List<Future<T>> invokeAll(Collection<? extends Callable<T>> tasks, long timeout, TimeUnit unit) ``` Let's see what Jun thinks. He will be the one committing probably.
54. **body:** @ijuma I understand the change may be better. But I have just spent 7 days fixing an issue that blocks the real patch (#1005) I would like to get in. I have a lot more to work on before 0.10 and would like to spend my time there. This code works, it proves the fix works, and its test code...so we can always make it better.
    label: code-design
55. **body:** @junrao @granthenke agreed that fixing zkclient is the ideal option to make it all uniform. I was thinking that even with zkclient you can get a connection timeout exception or whatever it is called because the client might not be able to reconnect in a long time. Consequently, the caller of updateResourceAcls needs to be ready to either recover or retry in the case the update doesn't succeed. If we get a connection loss event, then we could just retry. If we get a session expired, then we need to propagate the exception up perhaps wrapping it with some other kafka exception so that we don't expose session expiration. The zkclient instance will session as Jun says, so if the caller retries and zkclient is able to create the new session, then we will be able to perform the delete. We can try to have zkclient fixed, we did it for 0.9 at the last minute and it worked, so it might be doable. Otherwise, I think the workaround isn't bad, but if we go with this workaround, then we should have a jira for Kafka as well so that we remember that it is there. Does it make sense?
    label: code-design

56. **body:** This is going to be a style comment, so it is arguable whether it is better or not. I'd rather have this try/catch wrapping the call to `conditionalUpdatePersistentPathIfExists` because we are specifically interested in the exception that it can throw. I think it makes the intent more clear if we move it there, but again, up to you.
    label: code-design
57. @fpj I moved the update logic to its own private method. Pushing an update soon.
58. @ijuma I updated the code to use invokeAll. Please confirm its what you had in mind.
59. **body:** Looks like IntelliJ reformatted the code here (and a few other places in this file). It's quite annoying when it does that.
    label: code-design
60. Thanks! Yes, that's pretty close to what I had in mind. See the following PR for a few suggested improvements: https://github.com/granthenke/kafka/pull/2 If you agree, feel free to merge into your branch.
61. Minor point, but I'm wondering if it is best to have a call in ZkUtils so that we consistently access zkclient through ZkUtils.
62. **body:** @fpj I chose not to because it's not used anywhere else yet and I didn't want to think through and test "generic logic" for all uses. This code is contained and tested with respect to the authorizer. I thought was was an okay choice because ZkClient is used in many other places throughout the code base as well.
    label: code-design
63. **body:** I'm thinking that we should bound the number of iterations of the loop to be extra safe. It is unlikely that it happens, but if some reason it has to iterate over this loop many times, then it might be better to fail the operation.
    label: code-design
64. @fpj Do you have a suggest number of tries we should limit this too? Under what scenarios do you think it would loop indefinitely without some other exception from zookeeper breaking the loop?
65. Just to confirm one thing, you want to check here if the new code does not throw any exception under concurrent requests, you're not checking the previous incorrect behavior in which authorizers are overwriting the znode data of each other, is that right? If so, this is good, but it is not really testing race conditions like you spotted originally in the jira.
66. If you want to make more evident what the lock is covering, I think you can move it inside the if block. This observation also holds for other lock blocks in the patch.
67. @fpj This is a generic test utility to test concurrent code. This piece will only fail under exceptions caused by the concurrent operations. The test using this code still needs to validate the end state is as expected. I use this code in `SimpleAclAuthorizerTest.testHighConcurrencyModificationOfResourceAcls` and validate the state (that we are not dropping acls) after this is called.
68. **body:** I don't think there is a right answer here for the number of retries, and it should really never happen that you're looping indefinitely other than getting a bad version on every attempt. But, it does feel safer to have a safeguard for the case that something goes wrong. If you want a guess, I'd say 5 times but I'm happy if you want to go higher.
    label: code-design
69. **body:** The change isn't strictly necessary, and in fact it makes it less efficient because we have an additional call. I was thinking about the case that we want to replace zkclient with direct calls to the raw zookeeper client (KAFKA-3210) and as I was thinking about doing it, we'd just rewire ZkUtils. It's not a big deal if you don't want to change, though. Just let me know so that we can wrap this one up.
    label: code-design
70. @fpj I will set this to 15 to be "safe" but ensure its not triggered too soon. I started with 10 but my "high concurrency" test actually triggered it and failed.
71. @fpj I can move it.

jira_issues:

1. **summary:** SimpleAclAuthorizer can lose ACLs with frequent add/remove calls
   description: Currently when adding or removing an ACL with the SimpleAclAuthorizer the following high level steps happen: # read acls from cache # merge with the changes acls # update zookeeper # add a change notification Then the Authorizers listening for the change notification know to invalidate their cache and get the latest value. However that takes some time. In the time between the ACL change and the cache update, a new add or remove request could be made. This will follow the steps listed above, and if the cache is not correct all changes from the previous request are lost. This can be solved on a single

node, by updating the cache at the same time you update zookeeper any time a change is made. However, because there can be multiple instances of the Authorizer, a request could come to a separate authorizer and overwrite the Zookeeper state again loosing changes from earlier requests. To solve this on multiple instances. The authorizer could always read/write state from zookeeper (instead of the cache) for add/remove requests and only leverage the cache for get/authorize requests. Or it could block until all the live instances have updated their cache. Below is a log from a failed test in the WIP [pull request|https://github.com/apache/kafka/pull/1005] for KAFKA-3266 that shows this behavior: {noformat} [2016-03-03 11:09:20,714] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Allow permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,726] DEBUG updatedAcls: Set(User:ANONYMOUS has Allow permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,738] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Deny permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,739] DEBUG updatedAcls: Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,752] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,755] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,762] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,768] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,773] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,777] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) {noformat}
2. **summary:** SimpleAclAuthorizer can lose ACLs with frequent add/remove calls
description: Currently when adding or removing an ACL with the SimpleAclAuthorizer the following high level steps happen: # read acls from cache # merge with the changes acls # update zookeeper # add a change notification Then the Authorizers listening for the change notification know to invalidate their cache and get the latest value. However that takes some time. In the time between the ACL change and the cache update, a new add or remove request could be made. This will follow the steps listed above, and if the cache is not correct all changes from the previous request are lost. This can be solved on a single node, by updating the cache at the same time you update zookeeper any time a change is made. However, because there can be multiple instances of the Authorizer, a request could come to a separate authorizer and overwrite the Zookeeper state again loosing changes from earlier requests. To solve this on multiple instances. The authorizer could always read/write state from zookeeper (instead of the cache) for add/remove requests and only leverage the cache for get/authorize requests. Or it could block until all the live instances have updated their cache. Below is a log from a failed test in the WIP [pull request|https://github.com/apache/kafka/pull/1005] for KAFKA-3266 that shows this behavior: {noformat} [2016-03-03 11:09:20,714] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Allow permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,726] DEBUG updatedAcls: Set(User:ANONYMOUS has Allow permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,738] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Deny permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,739] DEBUG updatedAcls: Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,752] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,755] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,762] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03

11:09:20,768] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,773] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,777] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) {noformat}

3. **summary:** SimpleAclAuthorizer can lose ACLs with frequent add/remove calls
   description: Currently when adding or removing an ACL with the SimpleAclAuthorizer the following high level steps happen: # read acls from cache # merge with the changes acls # update zookeeper # add a change notification Then the Authorizers listening for the change notification know to invalidate their cache and get the latest value. However that takes some time. In the time between the ACL change and the cache update, a new add or remove request could be made. This will follow the steps listed above, and if the cache is not correct all changes from the previous request are lost. This can be solved on a single node, by updating the cache at the same time you update zookeeper any time a change is made. However, because there can be multiple instances of the Authorizer, a request could come to a separate authorizer and overwrite the Zookeeper state again loosing changes from earlier requests. To solve this on multiple instances. The authorizer could always read/write state from zookeeper (instead of the cache) for add/remove requests and only leverage the cache for get/authorize requests. Or it could block until all the live instances have updated their cache. Below is a log from a failed test in the WIP [pull request|https://github.com/apache/kafka/pull/1005] for KAFKA-3266 that shows this behavior: {noformat} [2016-03-03 11:09:20,714] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Allow permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,726] DEBUG updatedAcls: Set(User:ANONYMOUS has Allow permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,738] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Deny permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,739] DEBUG updatedAcls: Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,752] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,755] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,762] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,768] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,773] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,777] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) {noformat}

4. **summary:** SimpleAclAuthorizer can lose ACLs with frequent add/remove calls
   description: Currently when adding or removing an ACL with the SimpleAclAuthorizer the following high level steps happen: # read acls from cache # merge with the changes acls # update zookeeper # add a change notification Then the Authorizers listening for the change notification know to invalidate their cache and get the latest value. However that takes some time. In the time between the ACL change and the cache update, a new add or remove request could be made. This will follow the steps listed above, and if the cache is not correct all changes from the previous request are lost. This can be solved on a single node, by updating the cache at the same time you update zookeeper any time a change is made. However, because there can be multiple instances of the Authorizer, a request could come to a separate authorizer and overwrite the Zookeeper state again loosing changes from earlier requests. To solve this on multiple instances. The authorizer could always read/write state from zookeeper (instead of the cache) for add/remove requests and only leverage the cache for get/authorize requests. Or it could block until all the live instances have updated their cache. Below is a log from a failed test in the WIP [pull request|https://github.com/apache/kafka/pull/1005] for KAFKA-3266 that shows this behavior:

{noformat} [2016-03-03 11:09:20,714] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Allow permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,726] DEBUG updatedAcls: Set(User:ANONYMOUS has Allow permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,738] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Deny permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,739] DEBUG updatedAcls: Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,752] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,755] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,762] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,768] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,773] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,777] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) {noformat}

5. **summary:** SimpleAclAuthorizer can lose ACLs with frequent add/remove calls
description: Currently when adding or removing an ACL with the SimpleAclAuthorizer the following high level steps happen: # read acls from cache # merge with the changes acls # update zookeeper # add a change notification Then the Authorizers listening for the change notification know to invalidate their cache and get the latest value. However that takes some time. In the time between the ACL change and the cache update, a new add or remove request could be made. This will follow the steps listed above, and if the cache is not correct all changes from the previous request are lost. This can be solved on a single node, by updating the cache at the same time you update zookeeper any time a change is made. However, because there can be multiple instances of the Authorizer, a request could come to a separate authorizer and overwrite the Zookeeper state again loosing changes from earlier requests. To solve this on multiple instances. The authorizer could always read/write state from zookeeper (instead of the cache) for add/remove requests and only leverage the cache for get/authorize requests. Or it could block until all the live instances have updated their cache. Below is a log from a failed test in the WIP [pull request|https://github.com/apache/kafka/pull/1005] for KAFKA-3266 that shows this behavior: {noformat} [2016-03-03 11:09:20,714] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Allow permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,726] DEBUG updatedAcls: Set(User:ANONYMOUS has Allow permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,738] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Deny permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,739] DEBUG updatedAcls: Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,752] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,755] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,762] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,768] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,773] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,777] DEBUG Processing ACL change notification for Cluster:kafka-cluster and

Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) {noformat}

6. **summary:** SimpleAclAuthorizer can lose ACLs with frequent add/remove calls
   description: Currently when adding or removing an ACL with the SimpleAclAuthorizer the following high level steps happen: # read acls from cache # merge with the changes acls # update zookeeper # add a change notification Then the Authorizers listening for the change notification know to invalidate their cache and get the latest value. However that takes some time. In the time between the ACL change and the cache update, a new add or remove request could be made. This will follow the steps listed above, and if the cache is not correct all changes from the previous request are lost. This can be solved on a single node, by updating the cache at the same time you update zookeeper any time a change is made. However, because there can be multiple instances of the Authorizer, a request could come to a separate authorizer and overwrite the Zookeeper state again loosing changes from earlier requests. To solve this on multiple instances. The authorizer could always read/write state from zookeeper (instead of the cache) for add/remove requests and only leverage the cache for get/authorize requests. Or it could block until all the live instances have updated their cache. Below is a log from a failed test in the WIP [pull request|https://github.com/apache/kafka/pull/1005] for KAFKA-3266 that shows this behavior: {noformat} [2016-03-03 11:09:20,714] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Allow permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,726] DEBUG updatedAcls: Set(User:ANONYMOUS has Allow permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,738] DEBUG [KafkaApi-0] adding User:ANONYMOUS has Deny permission for operations: Describe from hosts: * for Cluster:kafka-cluster (kafka.server.KafkaApis:52) [2016-03-03 11:09:20,739] DEBUG updatedAcls: Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,752] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,755] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,762] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,768] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,773] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) [2016-03-03 11:09:20,777] DEBUG Processing ACL change notification for Cluster:kafka-cluster and Set(User:ANONYMOUS has Deny permission for operations: Describe from hosts: *) (kafka.security.auth.SimpleAclAuthorizer:52) {noformat}

jira_issues_comments:

1. I think I will work on this as it is affecting my KIP-4 work. Any thoughts on only using the cache for read operations and always communicating via Zookeeper for write operations?
2. Sounds good, however can we do lazy writes to ZK? Basically, batch multiple acls CRUD in a single ZK write? Depending on use-case, it will be helpful.
3. [~parth.brahmbhatt] might have some thoughts as well.
4. GitHub user granthenke opened a pull request: https://github.com/apache/kafka/pull/1006 KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remov… …e calls Changes the SimpleAclAuthorizer to: - Always read state from Zookeeper before updating acls - Update local cache when modifying acls You can merge this pull request into a Git repository by running: $ git pull https://github.com/granthenke/kafka simple-authorizer-fix Alternatively you can review and apply these changes as the patch at: https://github.com/apache/kafka/pull/1006.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #1006 ---- commit 3483b759e1cf5e5a42ad3206eca41e8e75906b41 Author: Grant Henke <granthenke@gmail.com> Date: 2016-03-03T20:59:03Z KAFKA-3328: SimpleAclAuthorizer can lose ACLs with frequent add/remove calls Changes the SimpleAclAuthorizer to: - Always read state from Zookeeper before updating acls - Update local cache when modifying acls ----
5. Issue resolved by pull request 1006 [https://github.com/apache/kafka/pull/1006]

6. Github user asfgit closed the pull request at: https://github.com/apache/kafka/pull/1006