算法2. 热度转换.

1. pg.changeFlag←false

2. if　addedHeat＞0　then

3. if pg.state＝cold　then

4. pg.state←warm；

5. else if　pg.state＝warm　then

6. pg.state←hot；

7. pg.changeFlag←true；

8. end if

9. else if　addedHeat＜0　then

10. if　pg.state＝hot then

11. pg.state←warm；

12. else if　pg.state＝warm then

13. pg.state←cold；

14. pg.changeFlag←true;

15. end if

16. end if

算法3. 倾向算法.

trendCalculate(Page pg)

1. flag←true //the flag of whether trend is reduced;

2. q←１－(pg.lr＋lg.lw)／pg.totalaccess;

3. //calculate the I/O cost on SSD and HDD

4. ssdcost←(pg. lr · q＋pg. pr) · rs+ (pg.lw · q＋pg.pw) · ws;

5. hddcost←(pg.lr · q＋pg.pr) · rh+ (pg.lw · q＋pg.pw) · wh;

6. //calculate the trend and its reduce

7. if　pg.state＝hot then

8. if　pg.changeFlag＝true then

9. pg.trend←(ssdcost-hddcost)＋pg.pretrend；

10. pg.pretrend←pg.trend · β；

11. pg.changeFlag←false；flag←true；

12. else

13. 　pg.trend←(ssdcost-hddcost)＋pg.pretrend；

14. 　flag←false;

15. endif

16. else if　pg.state＝warm then

17. pg.trend←(ssdcost－hddcost)＋pg.pretrend；

18. flag←false；

19. else if　pg.state＝cold then

20. if　pg.changeFlag＝true then

21. pg.trend←(ssdcost-hddcost)＋pg.pretrend；

22. 　pg.pretrend←pg.trend · β；

23. 　pg.changeFlag←false；flag←true；

24. else

25. interval←pg.cold2－pg.cold1；

26. if　interval／ssdSize＜hddSize／ssdSize then

27. pg.trend←(ssdcost-hddcost)＋pg.pretrend；

28. flag←false；

29. else

30. pg.pretrend←pg.trend.β · (ssdSize／interval)；

31. pg.trend←(ssdcost-hddcost)；

32. flag←true；

33. end if

34. end if

35. end if

36. //reset the counter

37. if　flag＝true then

38. pg.lr←0；pg.lw←0；pg.pr←0；pg.pw←0;

39. end if

40. if ｜pg.trend｜＞ws＋wh and pg.trend＜0 then

41. return SSD-trend;

42. else if ｜pg.trend｜＞ws＋wk and pg.trend＞0

43. return HDD-trend;

44. else

45. return pg.placement;

46. end if

算法4. 混合存储算法.

accessPage(Page pg)

1. pg.totalaccess＋＋；

2. if　pg is found in buffer then

3. move pg to the head of buffer LRU；

4. if　access is read quest then

5.　 pg.lr＋＋；

6.　 else if　access is write quest then

7. pg.lw＋＋；

8. end if

9. else

10. if　buffer is full then

11. evictPage();

12. end if

13. fetch pg from disk;

14. updateState(pg);

15. add pg to the head of buffer LRU;

16. if　access is read quest then

17. pg.pr＋＋；

18. else　access is write quest then

19. pg.pw＋＋；

20. end if

21. end if

算法5. evictPage().

1. Page pg；

2. pg←tail of LRU；

3. place←trendCalculate(pg)；

4. if　pg is on HDD then

5. if　pg.state is hot or warm and place＝SSD-trend

6. migratePage(pg)；

7. end if

8. else if　pg is on SSD then

9. if　place = SSD-trend then

10. migratePage(pg)；

11. else if　pg is cold and SSD write performance is beyond HDD then

12. migratePage(pg)；

13. end if

14. end if

15. if　pg is dirty then

16. write pg to disk of pg.placement；

17. end if