kubernetes #78782 issue : #78776

commit ece3e3cdba4985215d76c82ba713a778fe978aaf

问题： 访问节点的逻辑不正确

原因：

NodeTree的Next方法会对NodeTree加锁，造成顺序访问NodeTree锁竞争严重，从而性能下降。开发人员对此进行了优化，使用allNodes来记录一个slice，利用lastIndex和len(allNodes)来访问allNodes来访问NodeTree的节点，但是由于NodeTree节点是动态改变的，所以在更新lastIndex时，len(allNodes)可能为零，导致了模零错误。

```
@@ -462,8 +461,8 @@ func (g *genericScheduler) findNodesThatFit(pod *v1.Pod, nodes
[]*v1.Node) ([]*v
        if len(g.predicates) == 0 {
                filtered = nodes
        } else {
-               allNodes := g.cache.NodeTree().AllNodes()
-               numNodesToFind := g.numFeasibleNodesToFind(int32(len(allNodes)))
+               allNodes := int32(g.cache.NodeTree().NumNodes())
+               numNodesToFind := g.numFeasibleNodesToFind(allNodes)

                // Create filtered list with enough space to avoid growing it
                // and allow assigning.
@@ -479,12 +478,8 @@ func (g *genericScheduler) findNodesThatFit(pod *v1.Pod, nodes
[]*v1.Node) ([]*v
                // We can use the same metadata producer for all nodes.
                meta := g.predicateMetaProducer(pod, g.nodeInfoSnapshot.NodeInfoMap)

-               processedNodes := int32(0)
                checkNode := func(i int) {
-                       // We check the nodes starting from where we left off in the
previous scheduling cycle,
-                       // this is to make sure all nodes have the same chance of being
examined across pods.
-                       atomic.AddInt32(&processedNodes, 1)
-                       nodeName := allNodes[(g.lastIndex+i)%len(allNodes)]
+                       nodeName := g.cache.NodeTree().Next()
                        fits, failedPredicates, err := podFitsOnNode(
                                pod,
                                meta,
@@ -516,8 +511,7 @@ func (g *genericScheduler) findNodesThatFit(pod *v1.Pod, nodes
[]*v1.Node) ([]*v

                // Stops searching for more nodes once the configured number of feasible
nodes
                // are found.
-               workqueue.ParallelizeUntil(ctx, 16, len(allNodes), checkNode)
-               g.lastIndex = (g.lastIndex + int(processedNodes)) % len(allNodes)
```

```
+               workqueue.ParallelizeUntil(ctx, 16, int(allNodes), checkNode)

                filtered = filtered[:filteredLen]
                if len(errs) > 0 {
```

修复：

回复原来的性能较低的实现。

触发：

扩展CrashTuner->在len(allNodes)-->多个crash

影响：

cluster down