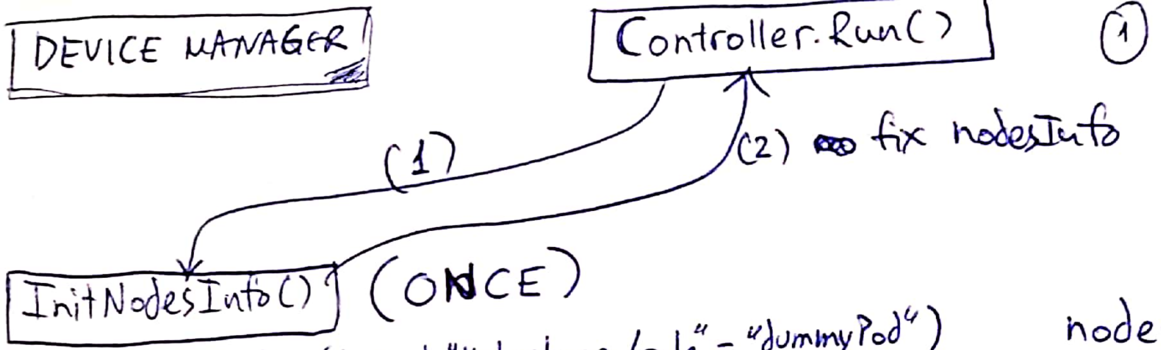
# DEVICE MANAGER



```
pods = dummy pods (label "kubeshare/role" = "dummyPod")
sharepods = all SharePods on the apiserver
- for pod in pods (dummypods) : {
      GPUID = labels ("kubeshare/GPUID")
      if NodesInfo[nodeName] DOESN'T EXIST {
            make nodesInfo[nodeName] with
                 empty fields
  A  [  make nodesInfo[NodeName].GPUID2GPU[GPUID]
       [      and set default fields : UUID = ""
       [                                Usage = 0.0
                                        Mem = 0
                                        PodList = empty list
      }
      else (if nodeInfo[nodeName] EXISTS, just do A.
- for sharepod in sharepods : {
    check Annotations (gpu-request, limit, mem) make sense (< 1.0, non-negative, etc)
    if nodesInfo[nodeName].GPUID2GPU[GPUID] exists : {
                       add PodRequest to PodList of GPUID

    fix processDummyPodLater (this is in case device
                     manager is restarted
                     while Pods are still running)
```

```
nodesInfo = map [nodeName] → NodeInfo

NodeInfo = struct {
      GPUID2GPU = map[GPUID] → GPUInfo
            struct {UUID, Usage, Mem,
                    PodList} ⊛
      UUID2Port = map[UUID] → string
      PodManagerPortBitMap   bitmap
      PodIP
}
⊛ PodList = List of struct PodRequest

   struct PodRequest : {
        Key : string
        Request : float
        Limit : float
        Memory : int
        PodManagerPort : int
```

Controller.Run() ②

StartConfigManager()

server that talks to config-client on the nodes:

(1) receives list of UUIDs of ~~all~~ ALL GPUs on the Node from config-client (establishes a UUID Database)

(2) update Annotations of NodeStatus on the apiserver
"Kubeshare/gpu-info" = GPU-UUID : mem(in MiB)

(3) update UUID2Port in nodesInfo[nodeName]
+ PodIp

(4) syncConfig() : send PodList of that Node to config-client for every GPUID that belongs to that node

config-client :

check notebook Notes !!

- sends hostname, list of UUIDs
- waits for "Requests", writes PodList to special files.

handle_object()

For Pods (dummy).
extracts UUID from their logs and updates
nodesInfo[nodeName].
GPUID2GPU[GPUID].UUI

Controller.Run() (3)

processNextWorkItem() [for sharePods on the apiserver]

↓

syncHandler()

Input is a sharepod that has been Scheduled (nodeName ≠ "")

if node.GPUID2GPU[GPUID] exists:
add PodRequest {Key ~~Req~~, Limit, Memory, Port}
to the ~~pod~~ GPU's PodList
and call syncConfig()

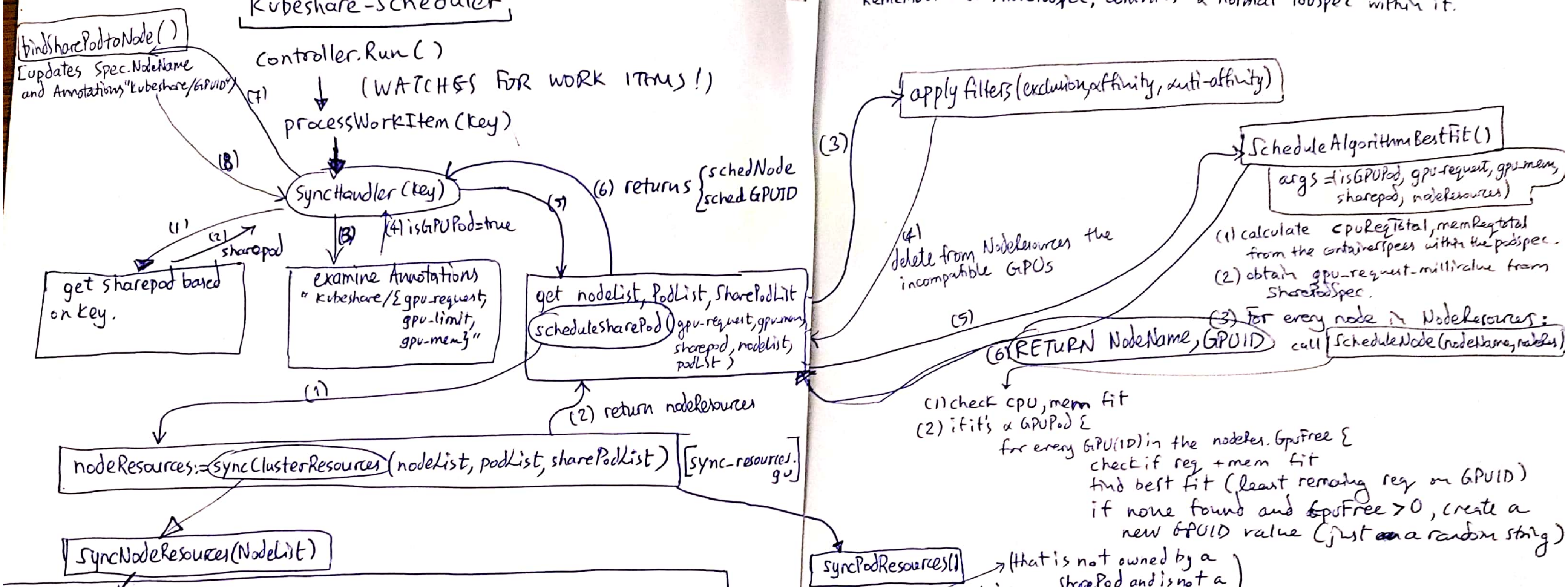
if it doesn't exist (UUID
unknown

- add PodRequest to ~~8~~ gpu-PodListE
- createDummyPod (nodeName, GPUID)

when dummypod gets created:
(getobject)updates UUID of
nodesInfo)
the Key of PodRequest is used
to requeue it ~~foo~~ for
processing

# Kubeshare-scheduler

Remember: a SharePodSpec, contains a normal PodSpec within it.



bindSharePodtoNode()
[updates Spec.NodeName
and Annotations "kubeshare/GPUID"]

controller.Run()
(WATCHES FOR WORK ITEMS!)

processWorkItem(key)

SyncHandler(key)

(1) (2) sharepod

get sharepod based on key.

(3)
(4) isGPUPod=true

examine Annotations
"kubeshare/{gpu-request,
gpu-limit,
gpu-mem}"

(5)
(6) returns {schedNode
schedGPUID

(7)
(8)

get nodeList, PodList, SharePodList
scheduleSharePod() (gpu-request, gpu-mem, sharepod, nodelist, podList)

(1)
(2) return nodeResources

nodeResources := syncClusterResources(nodeList, podList, sharePodList) [sync-resources.go]

syncNodeResources(NodeList)

(3)
apply filters (exclusion, affinity, anti-affinity)

(4)
delete from NodeResources the incompatible GPUs

(5)

ScheduleAlgorithmBestFit()
args = (isGPUPod, gpu-request, gpu-mem, sharepod, nodeResources)

(1) calculate cpuReqTotal, memReqtotal from the containerSpecs within the podspec.
(2) obtain gpu-request-millivalue from SharePodSpec.
(3) For every node in NodeResources: call ScheduleNode(nodeName, nodeRes)
(6) RETURN NodeName, GPUID

```
(1) check cpu, mem fit
(2) if it's a GPUPod {
        for every GPU(ID) in the nodeRes.GpuFree {
            check if req + mem fit
            find best fit (least remaing req on GPUID)
            if none found and GpuFree > 0, create a
                new GPUID value (just a random string)
```

nodeResources is a map[nodeName] → struct NodeResource

```
struct NodeResource {
        CpuTotal        int64
        MemTotal        int64
        GpuTotal        int
        GpuMemTotal     int64
        CpuFree         int64
        MemFree         int64
        GpuFreeCount    int
        GpuFree  map[GPUID] → GPUInfo
                               ↳ struct { GPUFreeReq
                                          GPUFreeMem
                                          GPUAffinityTags []
                                          GPUAntiAffinityTags []
                                          GPUExclusionTags []
```

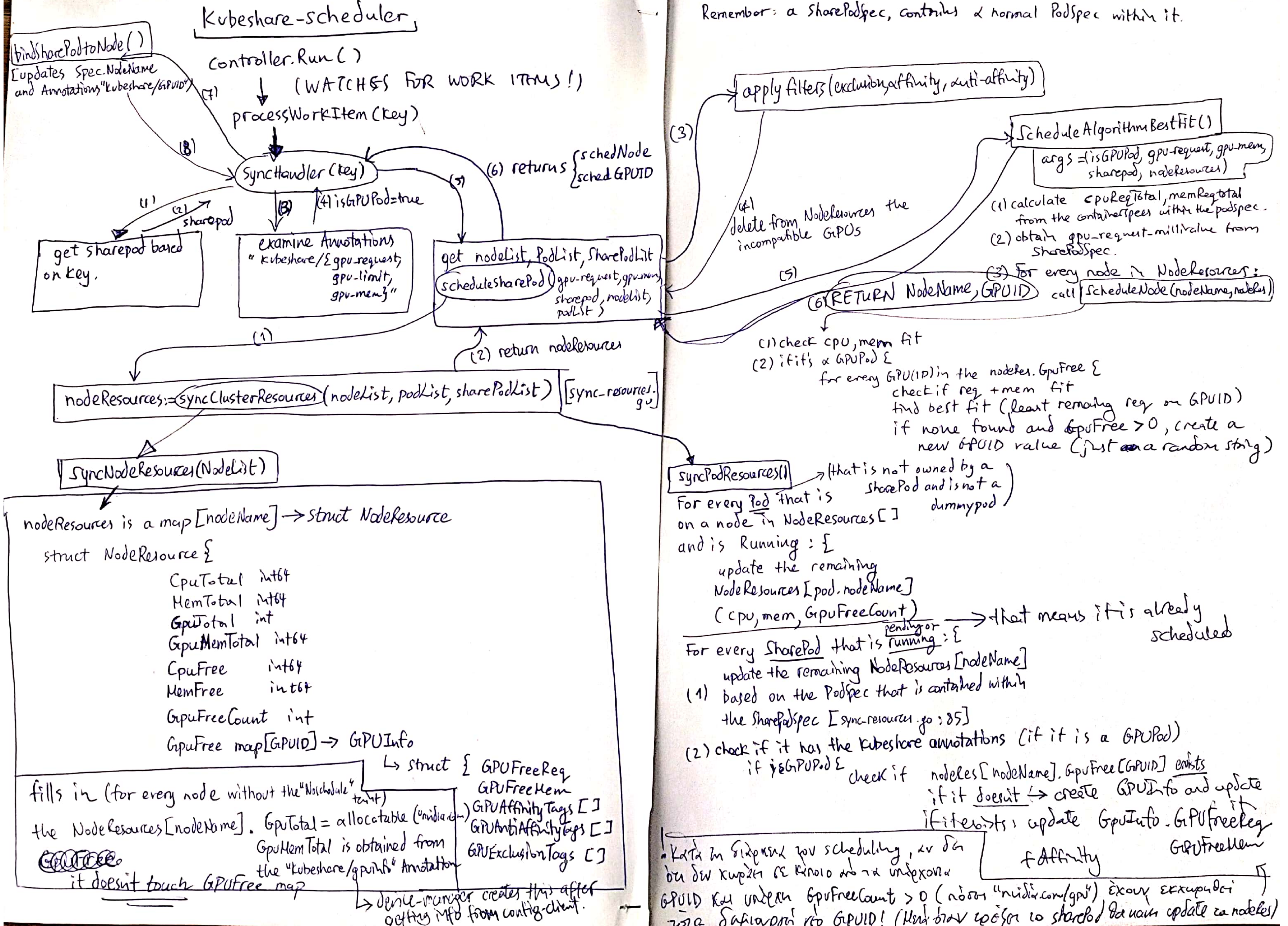
fills in (for every node without the "NoSchedule" taint)
the NodeResources[nodeName]. GpuTotal = allocatable ("nvidia.com")
GpuMemTotal is obtained from
the "kubeshare/gpuinfo" Annotation
↳ device-manager creates this after getting info from config-client.

it doesn't touch GPUFree map

syncPodResources()
For every Pod that is (that is not owned by a SharePod and is not a dummypod)
on a node in NodeResources[]
and is Running: {
update the remaining
NodeResources[pod.nodeName]
(cpu, mem, GpuFreeCount)

For every SharePod that is pending or running: { → that means it is already scheduled
update the remaining NodeResources[nodeName]
(1) based on the PodSpec that is contained within the SharePodSpec [sync-resources.go:85]
(2) check if it has the kubeshare annotations (if it is a GPUPod)
if isGPUPod {
check if nodeRes[nodeName].GpuFree[GPUID] exists
if it doesn't → create GPUInfo and update it
if it exists: update GpuInfo.GPUFreeReq
GPUFreeMem
+Affinity

• Κατά τη διάρκεια του scheduling, αν δει ότι δεν χωράει σε κάποιο από τα υπάρχοντα GPUID και υπάρχει GpuFreeCount > 0 (πόσοι "nvidia.com/gpu") έχουν εκχωρηθεί [illegible] GPUID! [illegible] update τα nodeRes,



Kubeshare-scheduler

controller.Run()

(WATCHES FOR WORK ITEMS!)

processWorkItem(key)

bindSharePodtoNode()

[updates Spec.NodeName and Annotations "kubeshare/GPUID"]

(7)

(8)

syncHandler(key)

(1)

(2) sharepod

get sharepod based on key.

(3)

(4) isGPUPod=true

examine Annotations " kubeshare/{gpu-request, gpu-limit, gpu-mem}"

(5)

(6) returns {schedNode schedGPUID

get nodeList, PodList, SharePodList

scheduleSharePod()(gpu-request, gpu-mem, sharepod, nodelist, podList)

(1)

(2) return nodeResources

nodeResources:=syncClusterResources(nodeList, podList, sharePodList)

[sync-resources.go]

syncNodeResources(NodeList)

nodeResources is a map[nodeName] → struct NodeResource

```
struct NodeResource {
        CpuTotal        int64
        MemTotal        int64
        GpuTotal        int
        GpuMemTotal     int64
        CpuFree         int64
        MemFree         int64
        GpuFreeCount    int
        GpuFree  map[GPUID] → GPUInfo
                               ↳ struct { GPUFreeReq
                                          GPUFreeMem
                                          GPUAffinityTags [ ]
                                          GPUAntiAffinityTags [ ]
                                          GPUExclusionTags [ ]
```

fills in (for every node without the "NoSchedule" taint) the NodeResources[nodeName]. GpuTotal = allocatable ("nvidia.com") GpuMemTotal is obtained from the "kubeshare/gpuinfo" Annotation ↳ device-manager creates this after getting info from config-client.

it doesnt touch GPUFree map

Remember: a SharePodSpec, contains a normal PodSpec within it.

(3)

apply filters (exclusion, affinity, anti-affinity)

(4) delete from NodeResources the incompatible GPUs

(5)

ScheduleAlgorithmBestFit()

args = (isGPUPod, gpu-request, gpu-mem, sharepod, nodeResources)

(1) calculate cpuReqTotal, memReqTotal from the containerSpecs within the podspec.

(2) obtain gpu-request-millivalue from SharePodSpec.

(3) for every node in NodeResources: call ScheduleNode(nodeName, nodeRes)

(6) RETURN NodeName, GPUID

```
(1) check cpu, mem fit
(2) if it's a GPUPod {
        for every GPU(ID) in the nodeRes.GpuFree {
                check if req + mem fit
                find best fit (least remaining req on GPUID)
                if none found and GpuFree > 0, create a
                   new GPUID value (just a random string)
```

syncPodResources() → (that is not owned by a SharePod and is not a dummypod)

```
For every Pod that is
on a node in NodeResources[ ]
and is Running : {
    update the remaining
    NodeResources[pod.nodeName]
    (cpu, mem, GpuFreeCount)
```

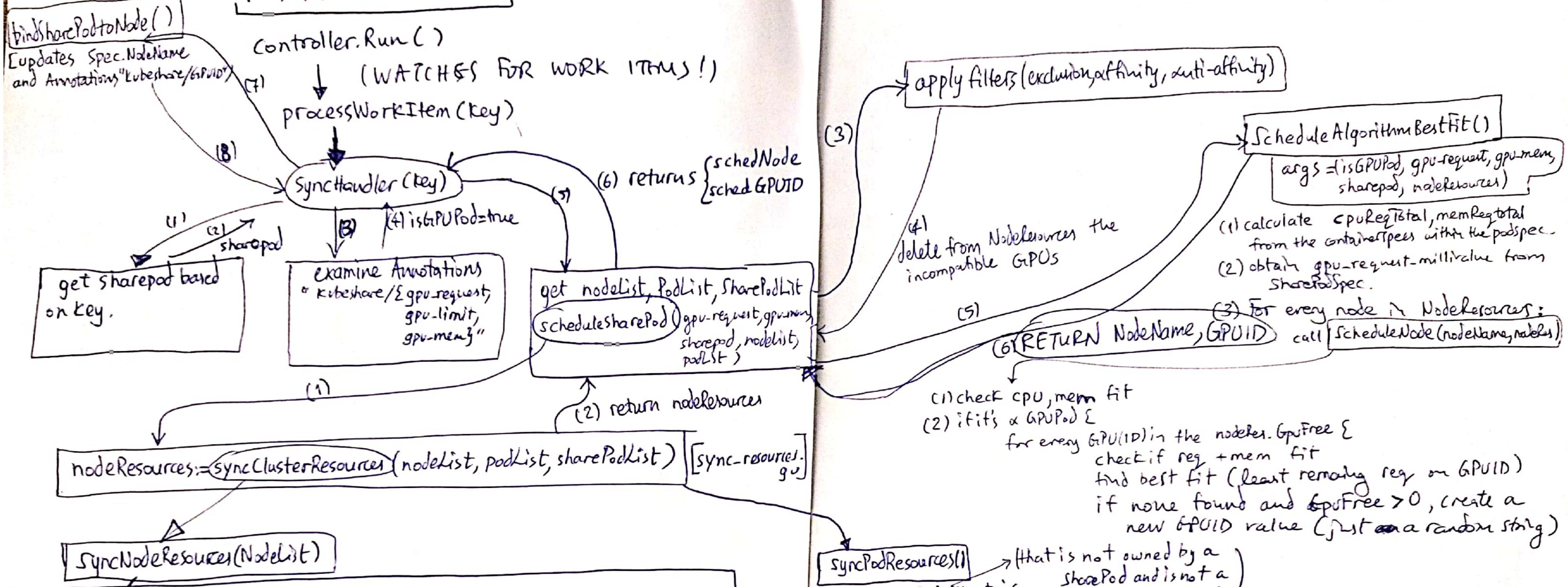
→ that means it is already scheduled

```
For every SharePod that is pending or running : {
    update the remaining NodeResources[nodeName]
(1) based on the PodSpec that is contained within
    the SharePodSpec [sync-resources.go:85]
(2) check if it has the kubeshare annotations (if it is a GPUPod)
        if isGPUPod {
                       check if  nodeRes[nodeName].GpuFree[GPUID] exists
                                 if it doesnt → create GPUInfo and update it
                                 if it exists: update GpuInfo.GPUFreeReq
                                                              GPUFreeMem
                                               + Affinity
                                                              }
```

• Κατά τη διάρκεια του scheduling, αν δει ότι δεν χωράει σε κάποιο από τα υπάρχοντα GPUID και υπάρχει GpuFreeCount > 0 (πόσοι "nvidia.com/gpu") έχουν εκχωρηθεί τότε δημιουργεί νέο GPUID! (Μετά όταν τρέξει το sharePod θα γίνει update τα nodeRes)

# Kubeshare-scheduler

Remember: a SharePodSpec, contains a normal PodSpec within it.


bindSharePodtoNode()
[updates Spec.NodeName and Annotations "kubeshare/GPUID"]
(7)
(8)
controller.Run()
(WATCHES FOR WORK ITEMS!)
processWorkItem(key)
SyncHandler(key)
(1)
(2) sharepod
get sharepod based on key.
(3)
(4) isGPUPod=true
examine Annotations
"kubeshare/{gpu-request, gpu-limit, gpu-mem}"
(5)
(6) returns {schedNode, schedGPUID
get nodeList, PodList, SharePodList
scheduleSharePod(gpu-request, gpu-mem, sharepod, nodelist, podList)
(1)
(2) return nodeResources
nodeResources := syncClusterResources(nodeList, podList, sharePodList)
[sync-resources.go]
syncNodeResources(NodeList)
syncPodResources()
(3)
apply filters (exclusion, affinity, anti-affinity)
(4) delete from NodeResources the incompatible GPUs
(5)
ScheduleAlgorithmBestFit()
args = (isGPUPod, gpu-request, gpu-mem, sharepod, nodeResources)
(6) RETURN NodeName, GPUID

(1) calculate cpuReqTotal, memReqtotal from the containerSpecs within the podspec.
(2) obtain gpu-request-millivalue from SharePodSpec.
(3) For every node in NodeResources: call ScheduleNode(nodeName, nodeRes)

```
(1) check cpu, mem fit
(2) if it's a GPUPod {
    for every GPU(ID) in the nodeRes.GpuFree {
        check if req + mem fit
        find best fit (least remaining req on GPUID)
        if none found and GpuFree > 0, create a
            new GPUID value (just ~~an~~ a random string)
```

syncNodeResources(NodeList)

nodeResources is a map[nodeName] → struct NodeResource

```
struct NodeResource {
        CpuTotal      int64
        MemTotal      int64
        GpuTotal      int
        GpuMemTotal   int64
        CpuFree       int64
        MemFree       int64
        GpuFreeCount  int
        GpuFree map[GPUID] → GPUInfo
                    ↳ struct { GPUFreeReq
                               GPUFreeMem
                               GPUAffinityTags [ ]
                               GPUAntiAffinityTags [ ]
                               GPUExclusionTags [ ]
```

fills in (for every node without the "NoSchedule" taint) the NodeResources[nodeName]. GpuTotal = allocatable ("nvidia.com/gpu") GpuMemTotal is obtained from the "kubeshare/gpuinfo" Annotation

↳ device-manager creates this after getting info from config-client.

it doesn't touch GPUFree map

syncPodResources()

For every Pod that is (that is not owned by a SharePod and is not a dummypod) on a node in NodeResources[ ] and is Running: {

update the remaining NodeResources[pod.nodeName] (cpu, mem, GpuFreeCount)

For every SharePod that is pending or running: { → that means it is already scheduled

update the remaining NodeResources[nodeName]

(1) based on the PodSpec that is contained within the SharePodSpec [sync-resources.go:85]

(2) check if it has the kubeshare annotations (if it is a GPUPod)

```
if isGPUPod {
    check if  nodeRes[nodeName].GpuFree[GPUID] exists
              if it doesn't → create GPUInfo and update it
              if it exists: update GpuInfo.GPUFreeReq
                                           GPUFreeMem
                            + Affinity
}
```

! Κατά τη διάρκεια του scheduling, αν δει ότι δεν χωράει σε κάποιο από τα υπάρχοντα GPUID και υπάρχει GpuFreeCount > 0 (πόσοι "nvidia.com/gpu" έχουν εκχωρηθεί) τότε δημιουργεί νέο GPUID! (Μετά όταν τρέξει το sharePod θα γίνει update τα nodeRes)