

PRIVACY COMPUTING TASK EXECUTION METHOD, PRIVACY TASK CONTROLLER, CENTRALIZED SCHEDULER

TECHNICAL FIELD

[0001] One or more embodiments of this specification relate to the field of computer technologies, and in particular, to a kubernetes cluster-based privacy computing task execution method, a privacy task controller in a kubernetes cluster, a task group controller in a kubernetes cluster, and a centralized scheduler in a kubernetes cluster.

BACKGROUND

[0002] Kubernetes is briefly referred to as k8s, and is an open-source and industrial container orchestration platform. Currently, emergence of k8s promotes popularization and landing of popular technologies such as a microservice architecture, so that development, operation and maintenance, and delivery become simpler. Although the standard k8s has provided many powerful functions, as application service scenarios become diversified, the standard k8s cannot satisfy a requirement, especially in some special application scenarios. Therefore, secondary development is started based on a k8s base.

[0003] Because a plurality of institutions usually need to jointly participate in a privacy computing task, how to ensure that a task can be efficiently scheduled between different institutions is a relatively popular research field in the industry. Currently, the standard k8s cannot satisfy a specific scenario of scheduling a multi-party privacy computing task. An embodiment of this specification provides a pioneering multi-party privacy computing task scheduling solution, to fill in a gap in this aspect.

SUMMARY

[0004] One or more embodiments of this specification describe a privacy computing task execution method, a privacy task controller, a task resource group controller, and a centralized scheduler, so that cross-institution pod scheduling can be implemented, scheduling is efficient, and

resource utilization is high.

[0005] According to a first aspect, a privacy computing task execution method is provided, related to a privacy task controller, a task resource group controller, and a centralized scheduler in a kubernetes cluster. The method includes:

5 creating, by the privacy task controller, a corresponding task resource group based on description information of a submitted privacy computing task, where information about the task resource group includes information about a plurality of participants of the privacy computing task; creating, by the privacy task controller in each of namespaces of the plurality of participants, several pods used to run the privacy computing task; learning, by the task resource group controller
10 through watching, that the task resource group is created, and creating a task resource in the task resource group in each of the namespaces of the plurality of participants, where information about the task resource includes a threshold of a quantity of pods in which a participant needs to reserve a resource; learning, by the centralized scheduler through watching, that any of the several pods is created, and updating a state of the task resource to a reserved state when it is determined that a
15 quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and determining, by the centralized scheduler based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

[0006] In an embodiment, the privacy task controller further creates another resource such as
20 a service discovery (Service) or application configuration management (configmap) in a namespace of each participant based on the description information of the privacy computing task.

[0007] In an embodiment, the information about the task resource group further includes a threshold of a quantity of task resources that have been in the reserved state, and the method further includes: updating, by the task resource group controller, each task resource in the task resource
25 group to a schedulable state if the task resource group controller determines that the quantity of task resources that have been in the reserved state in the task resource group is not less than the corresponding threshold of the quantity of task resources.

[0008] In a specific embodiment, the determining, by the centralized scheduler based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding
30 participant includes: learning, by the centralized scheduler through watching, that the state of the task resource is updated to the schedulable state, and scheduling the pod to the computing node of the corresponding participant if the quantity of pods in which a resource is reserved in the task

resource is not less than the corresponding threshold of the quantity of task resources.

[0009] In addition, in a specific embodiment, the information about the task resource further includes duration in which a participant reserves a resource, and after the state of the task resource is updated to the reserved state, the method further includes: if the centralized scheduler determines
5 that a timing reaches the duration and does not learn through watching that the reserved state of the task resource is updated to the schedulable state, releasing, by the centralized scheduler, a reserved resource of a corresponding participant, and/or changing the state of the task resource to "failed".

[0010] In an embodiment, the information about the task resource group further includes a
10 threshold of a quantity of task resources that have been in the reserved state, and the method further includes: if the task resource group controller determines that the quantity of task resources in the reserved state in the task resource group is less than the corresponding threshold of the quantity of task resources, determining, by the task resource group controller, whether a quantity of task resources in a reservable state and the reserved state in the task resource group is greater than or
15 equal to the threshold of the quantity of task resources; continuing waiting if it is determined that the quantity of task resources in the reservable state and the reserved state in the task resource group is greater than or equal to the threshold of the quantity of task resources; and if it is determined that the quantity of task resources in the reservable state and the reserved state in the task resource group is less than the threshold of the quantity of task resources, changing the task
20 resource group to a reserve failed state, and changing each task resource in the task resource group to a failed state.

[0011] In a specific embodiment, the information about the task resource group further includes rescheduling waiting duration; and after the changing the task resource group to a reserve failed state, and changing each task resource in the task resource group to a failed state, the method
25 further includes: after the waiting duration expires, separately changing, by the task resource group controller, states of the task resource group and each task resource to the reservable state, to enable a next round of scheduling for the privacy computing task.

[0012] In a specific embodiment, the information about the task resource group further includes a lifecycle of the task resource group; and the method further includes: updating, by the
30 task resource group controller, the state of the task resource group to "failed" when the task resource group controller determines that the lifecycle ends and scheduling of the privacy

computing task is not completed.

[0013] In a more specific embodiment, after the updating the state of the task resource group to "failed", the method further includes: after the privacy task controller learns through watching that the state of the task resource group is "failed", updating, by the privacy task controller, a task state of the privacy computing task to "failed", and/or deleting the task resource group and the several pods.

[0014] In an embodiment, the description information of the privacy computing task includes at least one of the following: the information about the plurality of participants, the threshold of the quantity of pods in which a participant needs to reserve a resource, or the threshold of the quantity of task resources that have been in the reserved state.

[0015] According to a second aspect, a privacy task controller in a kubernetes cluster is provided, including: a task resource group creation module, configured to create a corresponding task resource group based on description information of a submitted privacy computing task, where information about the task resource group includes information about a plurality of participants of the privacy computing task; and a pod creation module, configured to create, in each of namespaces of the plurality of participants, several pods used to run the privacy computing task. The kubernetes cluster further includes a task resource group controller, configured to: learn through watching that the task resource group is created, and create a task resource in the task resource group in each of the namespaces of the plurality of participants, where information about the task resource includes a threshold of a quantity of pods in which a participant needs to reserve a resource. The kubernetes cluster further includes a centralized scheduler, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

[0016] According to a third aspect, a task resource group controller in a kubernetes cluster is provided. The kubernetes cluster further includes a privacy task controller, configured to: create a corresponding task resource group based on description information of a submitted privacy computing task, where information about the task resource group includes information about a plurality of participants of the privacy computing task; and create, in each of namespaces of the

plurality of participants, several pods used to run the privacy computing task. The task resource group controller includes: a task resource creation module, configured to: learn through watching that the task resource group is created, and create a task resource in the task resource group in each of the namespaces of the plurality of participants, where information about the task resource
5 includes a threshold of a quantity of pods in which a participant needs to reserve a resource. The kubernetes cluster further includes a centralized scheduler, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and
10 determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

[0017] According to a fourth aspect, a centralized scheduler in a kubernetes cluster is provided. The kubernetes cluster further includes a privacy task controller, configured to: create a corresponding task resource group based on description information of a submitted privacy
15 computing task, where information about the task resource group includes information about a plurality of participants of the privacy computing task; and create, in each of namespaces of the plurality of participants, several pods used to run the privacy computing task. The kubernetes cluster further includes a task resource group controller, configured to: learn through watching that the task resource group is created, and create a task resource in the task resource group in each of
20 the namespaces of the plurality of participants information about the task resource includes a threshold of a quantity of pods in which a participant needs to reserve a resource. The centralized scheduler includes: a state updating module, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the
25 pod depends is not less than the corresponding threshold of the quantity of pods; and a scheduling module, configured to determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

[0018] According to a fifth aspect, a computer-readable storage medium is provided. The computer-readable storage medium stores a computer program, and when the computer program
30 is executed on a computer, the computer is enabled to perform the method in the first aspect.

[0019] According to a sixth aspect, a computing device is provided, and includes a memory

and a processor. The memory stores executable code, and when the processor executes the executable code, the method in the first aspect is implemented.

[0020] According to the privacy computing task execution method, the privacy task controller, the task resource group controller, and the centralized scheduler that are disclosed in the
5 embodiments of this specification, cross-institution pod scheduling can be performed in a multi-party privacy computing task execution scenario, and cross-institution scheduling efficiency and resource utilization are effectively improved.

BRIEF DESCRIPTION OF DRAWINGS

[0021] To describe the technical solutions in the embodiments of the present invention more
10 clearly, the following briefly describes the accompanying drawings required for describing the embodiments. Apparently, the accompanying drawings in the following description show merely some embodiments of the present disclosure, and a person of ordinary skill in the art can still derive other drawings from these accompanying drawings without creative efforts.

[0022] FIG. 1 is a schematic diagram illustrating a system architecture of scheduling a cross-
15 institution multi-party privacy computing task based on k8s, according to an embodiment;

[0023] FIG. 2 is a schematic flowchart illustrating a privacy computing task execution method, according to an embodiment of this specification;

[0024] FIG. 3 is a schematic diagram illustrating substeps included in an optional implementation of step S31 in FIG. 2, according to an embodiment of this specification;

[0025] FIG. 4 is a schematic diagram illustrating a module structure of a privacy task controller in a k8s cluster, according to an embodiment of this specification;

[0026] FIG. 5 is a schematic diagram illustrating a module structure of a task resource group controller in a k8s cluster, according to an embodiment of this specification; and

[0027] FIG. 6 is a schematic diagram illustrating a module structure of a centralized scheduler
25 in a k8s cluster, according to an embodiment of this specification.

DESCRIPTION OF EMBODIMENTS

[0028] The following describes the solutions provided in this specification with reference to the accompanying drawings.

[0029] As mentioned above, a plurality of privacy computing participants perform coordinated computing of a privacy computing task (or referred to as a privacy task in this specification), and a final computing result is obtained. For example, the privacy computing task can be a multi-party joint modeling task, each participant holds some training data, and finally each party obtains a secret sharing segment of a model parameter. Because cross-institution (or referred to as cross-participant) pod scheduling is not considered for a k8s native scheduler, a specific scenario of scheduling a multi-party privacy computing task cannot be satisfied. A pod is a minimum deployable computing unit that can be created and managed in k8s, and a service runs in a pod manner.

5

[0030] Based on the above-mentioned observation and analysis, embodiments of this specification disclose a k8s-based privacy computing task execution solution, so that cross-institution pod scheduling can be implemented, and efficient scheduling and high resource utilization are implemented.

[0031] FIG. 1 is a schematic diagram illustrating a system architecture of scheduling a cross-institution multi-party privacy computing task based on k8s, according to an embodiment. As shown in FIG. 1, an entire system architecture mainly includes two parts: a k8s management cluster on a central side and a k8s computing node on an institution side.

15

[0032] The k8s management cluster includes a task controller (Task Controller), a task resource group controller (Task Resource Group Controller), and a centralized scheduler (centralized scheduler) that are related to this embodiment of this specification. The privacy task controller is mainly configured to convert a submitted task into a resource supported by k8s. Currently, for each task, the privacy task controller correspondingly creates a resource such as a task resource group (Task Resource Group), a pod, or a service in k8s. Only two types of resources of the task resource group and the pod are related to task scheduling. The task resource group controller is responsible for a lifecycle of the privacy computing task. It should be understood that the task resource group controller can be disposed independently of the privacy task controller, or can be integrated with the privacy task controller. The centralized scheduler includes a k8s-based extension scheduling plug-in for the privacy computing task, and implements cross-institution scheduling of the privacy computing task by using the plug-in.

20

25

[0033] The k8s computing node is configured to run a pod service. Each institution can include a plurality of homogeneous or heterogeneous computing nodes.

30

[0034] The following describes a procedure of a cross-institution privacy computing task scheduling method in the system architecture shown in FIG. 1. FIG. 2 is a schematic flowchart illustrating a privacy computing task execution method, according to an embodiment of this specification. A shown method procedure relates to communication and interaction between a plurality of ends. In addition to the privacy task controller, the task resource group controller, and the centralized scheduler that are shown in FIG. 1, the plurality of ends further include a k8s API server. It should be understood that the API server is a core component of k8s, and provides interfaces such as CRUD and watch (watch) of various resource objects of k8s, and is a data bus and a data center of a k8s system.

10 [0035] As shown in FIG. 2, the method includes the following steps.

[0036] Step S20: The privacy task controller, the task resource group controller, and the centralized scheduler watch, through an interface provided by the k8s API server, a k8s resource on which the privacy task controller, the task resource group controller, and the centralized scheduler focus.

15 [0037] For example, the privacy task controller, the task resource group controller, and the centralized scheduler are self-started, and watch, by using a watching (List-Watch) mechanism provided by k8s, a specific k8s resource on which the privacy task controller, the task resource group controller, and the centralized scheduler focus.

[0038] Step S21: The k8s API server receives a submitted privacy computing task.

20 [0039] For submission of the privacy computing task, in an embodiment, a user can create a privacy task CR in a k8s cluster by using a k8s kubectl tool. It should be noted that one of extended resource types provided by Kubernetes is a CRD whose full name is a CustomResourceDefinition, that is, a custom resource, and a Kubernetes CR represents a specific instance of the Kubernetes CRD. In another embodiment, an application can create a privacy task CR in the k8s cluster by invoking the k8s API.

25 [0040] The following shows a specific example of a privacy task CR created in the k8s cluster, and describes content included in the privacy task CR with reference to this example. It can be understood that in actual applications, the privacy task CR can include all or a part of example content.

30 1 apiVersion: kuscia.secretflow/v1 //Name kuscia.secretflow and version v1 of a custom resource API

	2	kind: KusciaTask	//A kind of a custom resource is
		KusciaTask	
	3	metadata:	//Metadata of a custom resource
	4	name: kuscia-task-c8tv	//A name of a custom resource is kuscia-
5		task-c8tv	
	5	spec:	//Detailed information of a custom
		resource	
	6	initiator: alice-1001	//An initiator of a task is alice-1001
	7	scheduleConfig:	//Task scheduling configuration
10	8	minReservedMembers: 2	//A minimum quantity of participants is 2
	9	resourceReservedSeconds: 15	//Duration in which a participant can reserve
		a resource is 15s	
	10	retryDurationSeconds: 15	//Rescheduling waiting duration is 15s
	11	lifecycleSeconds: 600	//A lifecycle of a custom resource is 600s
15	12	parties:	//Participant information
	13	- domainID: alice-1001	//A namespace identifier of the 1 st participant
		is alice-1001	
	14	minReservedPods:1	//A minimum quantity of pods in which a
		resource is reserved in a case of alice-1001 is 1	
20	15	template:	//Template
	16	replicas: 1	//A quantity of replicates of a pod is 1
	17	spec:	//Detailed information of a pod
	18	containers:	//Container
	19	- name: secretflow	//A name of a container is secretflow
25	20	...	
	21	- domainID: bob-2001	//A namespace identifier of the 2 nd participant
		is bob-2001	
	22	minReservedPods: 2	//A minimum quantity of pods in which a
		resource is reserved in a case of bob-2001 is 2	
30	23	template:	
	24	replicas: 2	//A quantity of replicates of a pod is 2
	25	spec:	
	26	containers:	
	27	- name: secretflow	
35	28	...	

	29	- domainID: joke-3001	//A namespace identifier of the 3 rd participant
		is joke-3001	
	30	minReservedPods: 2	//A minimum quantity of pods in which a
		resource is reserved in a case of joke-3001 is 2	
5	31	template:	
	32	replicas: 3	//A quantity of replicates of a pod is 3
	33	spec:	
	34	containers:	
	35	- name: secretflow	
10	36	...	

[0041] With reference to the above-mentioned example, it can be learned that the privacy task CR can include the following several items of content:

[0042] (1) A name (name) is shown in Row 4 in the above-mentioned example, and represents a name of a privacy task.

15 **[0043]** (2) Privacy task scheduling configuration (scheduleConfig) is shown in Row 7 to Row 11 in the above-mentioned example.

[0044] Further, the task scheduling configuration can include the following parameters:

[0045] Minimum participant quantity (minReservedMembers): The minimum participant quantity is shown in Row 8 in the above-mentioned example. The minimum participant quantity represents a minimum quantity of participants that satisfy a reserved state, or a threshold of a quantity of task resources that satisfy a reserved state. If this condition is satisfied, the task resource group scheduler sets a state of a task resource CR in a namespace of each participant to "schedulable (Schedulable)", to advance a subsequent operation of binding task pods in the namespace of the participant. Otherwise, the entire task continues waiting.

25 **[0046]** Duration in which a participant reserves a resource (resourceReservedSeconds): The duration in which a participant can reserve a resource is shown in Row 9 in the above-mentioned example. The duration in which a participant can reserve a resource indicates a maximum time period in which each participant occupies a resource after the participant reserves a resource for a pod in a namespace of the participant. If a task does not satisfy a condition of the minimum participant quantity in a time specified by the parameter, a pod related to the task in the namespace of each participant releases a reserved resource for use by a subsequent task, thereby improving resource utilization.

30

[0047] Rescheduling waiting time (retryDurationSeconds): The rescheduling waiting time is shown in Row 10 in the above-mentioned example. The rescheduling waiting time indicates a waiting time for a next time of rescheduling after a task is not successfully scheduled within duration in which a participant reserves a resource. That is, if a previous time of scheduling of a task fails, the task is rescheduled based on the time specified by the parameter.

[0048] Lifecycle (lifecycleSeconds) of a task resource group: If scheduling of a task fails within a time specified by the parameter, the entire task is considered as "failed".

[0049] (3) Information about each participant of a task (parties) is shown in Row 12 to Row 36 in the above-mentioned example. Specifically, the information about each participant can include the following information items:

[0050] Domain identifier (domainID) of the participant: A domain (domain) corresponds to a namespace (Namespace) in the k8s cluster.

[0051] Threshold of a quantity of pods in which a participant needs to reserve a resource (minReservedPods): The threshold indicates a minimum quantity of pods in which a resource is reserved in a namespace of the participant. When this condition is satisfied, the centralized scheduler updates the state of the task resource CR of a pod in the namespace of the participant to the reserved state (Reserved).

[0052] Quantity of replicates (replicas): The quantity of replicates indicates a quantity of replicates of a pod of a task in a namespace of a participant.

[0053] An example description of content of a privacy task CR created in the k8s cluster is provided above. It can be understood that the k8s API server stores the created privacy task CR and provides a related service interface.

[0054] Step S22: The privacy task controller learns through watching that the privacy computing task is created, to obtain the privacy computing task.

[0055] Step S23: The privacy task controller parses the privacy computing task, to obtain description information of the privacy computing task. It should be understood that the privacy task controller can parse the obtained privacy task CR, to obtain content information included in the obtained privacy task CR.

[0056] Step S24: The privacy task controller creates, in the k8s cluster based on the description information of the privacy computing task, a k8s resource related to the privacy computing task. Specifically, step S24 can include substeps S241 and S242.

[0057] In step S241, the privacy task controller creates a task resource group corresponding to the privacy computing task. In view of this, references can be made to a task resource group CR shown in FIG. 1.

[0058] A specific example of a task resource group CR is shown below, and content included in the task resource group is described with reference to the example.

```

1    apiVersion: kuscia.secretflow/v1    //Name kuscia.secretflow and version v1 of
a custom resource API
2    kind: TaskResourceGroup              //A kind of a custom resource is
TaskResourceGroup
10   3    metadata:                        //Metadata of a custom resource
4    labels:                             //Label information of a custom resource
5        kuscia.secretflow/task-name: kuscia-task-c8tv
6    name: trg-cer7sc5fr3lg1vr7lu4g    //A name of a custom resource is trg-
cer7sc5fr3lg1vr7lu4g
15   7    spec:                           //Detailed information of a custom resource
8        initiator: alice-1001           //An initiator of a task is alice-1001
9        minReservedMembers: 2           //A minimum quantity of participants is 2
10       resourceReservedSeconds: 15     //Duration in which a participant can reserve a
resource is 15s
20   11    retryDurationSeconds: 15       //Rescheduling waiting duration is 15s
12    lifecycleSeconds: 600              //A lifecycle of a custom resource is 600s
13    parties:                           //Participant information
14        - domainID: alice-1001         //A namespace identifier of the 1st participant
is alice-1001
25   15    minReservedPods: 1             //A minimum quantity of pods in which a
resource is reserved in a case of alice-1001 is 1
16    pods:                             //Pod information
17        - name: secretflow-5b69cf6f7c-gsvtq    //Name of a pod is secretflow-
5b69cf6f7c-gsvtq
30   18        - domainID: bob-2001       //A namespace identifier of the 2nd
participant is bob-2001
19    minReservedPods: 2                 //A minimum quantity of pods in which a
resource is reserved in a case of bob-2001 is 2
20    pods:                             //Pod information

```

21 - name: secretflow-55bdmjgv4a-4x998
22 - name: secretflow-2ed1030c3d-cbzf9
23 - domainID: bob-3001 //A namespace identifier of the 3rd
participant is joke-3001

5 24 minReservedPods: 2 //A minimum quantity of pods in which a
resource is reserved in a case of joke-3001 is 2

 25 pods: //Pod information
 26 - name: secretflow-5c152d5fa4-csp8n
 27 - name: secretflow-3dae870b25-g67mq
10 28 - name: secretflow-75d5f497bb-k5v55
 29 status: //State information of a task resource group
 30 phase: Reserved//A state of a task resource group is "reserved".
 31 retryCount: 2 //A quantity of rescheduling times is 2
 32 conditions:
15 33 - lastUpdateTime: "2023-01-05 T07:36:35Z"
 34 status: "True"
 35 type: Reserved

[0059] With reference to the above-mentioned example, it can be learned that the task resource group can include the following several items of content:

20 **[0060]** (1) Label information (labels) of the task resource group: The label information is shown in Row 4 and Row 5. kuscia.secretflow/task-name in Row 5 indicates a name of a privacy task CR of the task resource group CR.

[0061] (2) A name (name) is shown in Row 6, and indicates the name of the task resource group.

25 **[0062]** (3) A minimum quantity of participants (minReservedMembers) is shown in Row 9, and indicates that the minimum quantity of participants that satisfy the reserved state is inherited from the privacy task CR. The task resource group controller completes coordinated scheduling between a plurality of participants (Parties) at a level of a task by using the condition.

[0063] (4) Duration in which a participant reserves a resource (resourceReservedSeconds) is shown in Row 10, and indicates that a maximum time in which a resource is reserved after a resource is reserved for a pod in the namespace of each participant is inherited from the privacy task CR.

[0064] (5) A rescheduling waiting time (retryDurationSeconds) is shown in Row 11, and

indicates that a waiting time of a next time of rescheduling after a task is not successfully scheduled within specified duration in which a participant reserves a resource is inherited from the private task CR.

[0065] (6) A lifecycle (lifecycleSeconds) of the task resource group is shown in Row 12, and indicates that a longest lifecycle in which scheduling of a task is not completed is inherited from the privacy task CR.

[0066] (7) Information about each participant of a task (parties) is shown in Row 13 to Row 28, and indicates the information about each participant of the task. Specifically, the information about each participant can include the following information items:

[0067] Domain identifier (domainID) of a participant: The domain identifier is a namespace identifier of the participant.

[0068] Threshold of a quantity of pods in which a participant needs to reserve a resource (minReservedPods): The threshold indicates a minimum quantity of pods in which a resource is reserved in a namespace of the participant.

[0069] Pod information (pods) of a participant: The pod information indicates a name of a pod of a task in a namespace of the participant.

[0070] (8) State information (status) of the task resource group is shown in Row 29 to Row 35.

[0071] It should be noted that specific content in the task resource group can be inherited from the privacy task CR and filled in by the privacy task controller, or can be filled with a default value if some information in the privacy task CR is default. In an example, for the minimum quantity of participants in the task resource group, if the information about the privacy task CR includes a corresponding specified value, the specified value is used as the minimum quantity of participants; otherwise, a quantity of all participants mentioned in the privacy task CR can be used as the minimum quantity of participants. In another example, for the duration in which a participant can reserve a resource and that is included in the task resource group, if the task description information includes corresponding specified duration (for example, 15s), the specified duration is used as a specific duration value; otherwise, default duration (for example, 20s) is used as a specific duration value.

[0072] In the above-mentioned descriptions, the privacy task controller can create a task resource group CR corresponding to the privacy computing task.

[0073] In step S242, the privacy task controller creates, in each of namespaces of a plurality

of participants, several task pods used to run the privacy computing task. It should be understood that "several" in this specification means one or more. In addition, "several" in several task pods is used to limit "task pod" rather than "task".

[0074] Specifically, the privacy task controller locates a corresponding namespace of a participant based on the namespace identifier included in the description information of the privacy task CR, and creates a corresponding quantity of task pods based on pod template information included in the description information of the namespace.

[0075] A specific example of a task pod is shown below, and content included in a pod resource is described with reference to the example.

```

10      1    apiVersion: v1                                //Version v1 of a resource
API
      2    kind: Pod                                       //A kind of a resource is pod
      3    metadata:
      4      labels:                                       //Label information of a
15 resource
      5      kuscia.secretflow/initiator: alice-1001
      6      kuscia.secretflow/task-resource: tr-cer7sc5fr3lg1vr7lu4g-12edfs
      7      kuscia.secretflow/task-resource-group: trg-cer7sc5fr3lg1vr7lu4g
      8      kuscia.secretflow/task-name: kuscia-task-c8tv
20      9      name: secretflow-55bdmjgv4a-4x998          //Name of a resource
      10     namespace: bob-2001                          //Namespace in which a
resource is located
      11     ownerReferences:                             //Owner reference
information of a pod resource
25      12     - apiVersion: kuscia.secretflow/v1
      13       blockOwnerDeletion: true
      14       controller: true
      15       kind: KusciaTask
      16       name: kuscia-task-c8tv
30      17       uid: 9df34703-5bf7-4ece-94f4-6ca04151ca7a
      18       ...
      19     spec:                                         //Detailed information of
a pod
      20     containers:                                   //Container information

```

21	name: secretflow	//A name of a container is
		secretflow
22	...	

[0076] With reference to the above-mentioned example, it can be learned that the pod resource can include the following several items of content:

[0077] (1) Label information (labels) is shown in Row 4 to Row 8, and indicates a label of the pod resource. Specifically, the label information can include the following several items:

kuscia.secretflow/initiator: kuscia.secretflow/initiator represents an initiator of the privacy computing task.

kuscia.secretflow/task-resource: kuscia.secretflow/task-resource represents a name of a task resource CR of the pod resource. It should be noted that, the label content corresponding to the item can be marked by the task resource group controller by locating a pod in a corresponding namespace based on name information of the pod after creating the task resource CR.

kuscia.secretflow/task-resource-group: kuscia.secretflow/task-resource-group indicates a name of the task resource group CR of the task resource CR.

kuscia.secretflow/task-name: kuscia.secretflow/task-name indicates a name of a privacy task CR of the pod resource.

[0078] (2) A name (name) is shown in Row 9, and indicates a name of a pod resource.

[0079] (3) A namespace (namespace) is shown in Row 10, and indicates a namespace in which a pod resource is located.

[0080] (4) Owner reference information (ownerReferences) is shown in Row 11 to Row 17, and indicates owner reference information of a pod resource.

[0081] In the above-mentioned descriptions, the privacy task controller can create one or more corresponding task pods in the namespace of each participant.

[0082] In an embodiment, in addition to substeps S241 and S242, step S24 can further include substep S243, and other resources such as a service discovery (Service) and application configuration management (configmap) are created in a namespace of each participant.

[0083] Resources such as a task resource group CR and a task pod that correspond to the privacy computing task and that are created by the privacy task controller in the k8s cluster are described above.

[0084] Step S25: The task resource group controller learns through watching that the task resource group is created, and obtains the task resource group.

[0085] Step S26: The task resource group controller parses the obtained task resource group, to obtain content information included in the task resource group controller.

[0086] Step S27: The task resource group controller creates a task resource of the task resource group in each of the namespaces of the plurality of participants. Specifically, the task resource group controller locates the namespace of each participant based on the information obtained by parsing the task resource group, and creates a corresponding task resource. It should be understood that the task resource is mainly used to describe detailed information of a corresponding participant.

[0087] A specific example of a created task resource CR is shown below, and content included in the task resource CR is described with reference to the example.

```
10      1    apiVersion: kuscia.secretflow/v1    //Name kuscia.secretflow and version v1
of a custom resource API
      2    kind: TaskResource                    //A kind of a custom resource is TaskResource
      3    metadata:                            //Metadata of a custom resource
      4    labels:                              //Label information of a custom resource
15      5      kuscia.secretflow/initiator: alice-1001
      6      kuscia.secretflow/task-resource-group: trg-cer7sc5fr3lg1vr7lu4g
      7      name:tr-cer7sc5fr3lg1vr7lu4g      //A name of a custom resource is tr-
cer7sc5fr3lg1vr7lu4g
      8      namespace: bob-2001                //A namespace of a custom resource is bob-
20    2001
      9    spec:                               //Detailed information of a custom resource
      10      initiator: alice-1001              //An initiator of a task is alice-1001
      11      minReservedPods: 2                //A minimum quantity of pods in which a
resource is reserved is 1
25      12      resourceReservedSeconds: 15    //Duration in which a participant can reserve a
resource is 15s
      13      pods:                            //Pod information
      14        - name: secretflow-55bdmjgv4a-4x998    //Name of a pod
      15        resources:                      //Resource information of a
30    pod
      16          limits:
      17            cpu: 1
      18            memory: 2Gi
      19          requests
```

```

20         cpu: 100m
21         memory: 200Mi
22     - name: secretflow-2ed1030c3d-cbzf9
23     resources:
5       24         limits:
25             cpu: 1
26             memory: 2Gi
27         requests:
28             cpu: 100m
10      29         memory: 200Mi
30     status: //State information of a task resource
31     phase: Reserved
32     conditions:
33     - lastUpdateTime: "2023-01-05 T07:36:35Z"
15      34     status: "True"
35     type: Reserved

```

[0088] With reference to the above-mentioned example, it can be learned that the task resource can include the following several items of content:

[0089] (1) Label information (labels) is shown in Row 4 to Row 6, and indicates a label of the task resource CR. kuscia.secretflow/task-resource-group in Row 5 indicates a name of a task resource group CR of a task resource CR.

[0090] (2) A name (name) is shown in Row 7, and indicates the name of the task resource CR.

[0091] (3) A namespace (namespace) is shown in Row 8, and indicates a namespace in which a task resource CR is located.

[0092] (4) A minimum quantity of pods in which a resource is reserved of a participant (minReservedPods) is shown in Row 11, and is inherited from the task resource group CR.

[0093] (5) Duration in which a participant can reserve a resource (resourceReservedSeconds) is shown in Row 12, and is inherited from the task resource group CR.

[0094] (6) Pod information (pods) is shown in Row 13 to Row 29, and indicates pod information included in the task resource CR.

[0095] (7) State information (status) is shown in Row 30 to Row 35, and indicates the state information of the task resource CR.

[0096] It should be noted that specific content of the task resource can be inherited from a task

resource group of the task resource, or can be obtained from a task pod of the task resource, or can be filled with a default value.

[0097] A task resource created by the task resource group controller in a namespace of each participant is described above.

5 **[0098]** Step S28: The centralized scheduler learns through watching that a task pod and a task resource are created, and processes the task pod and the task resource.

[0099] Step S29: The centralized scheduler determines whether a quantity of pods in which a participant reserves a resource in a task pod of the same task resource is not less than a threshold of a quantity of pods in the task resource (minReservedPods).

10 **[0100]** In an embodiment, the centralized scheduler parses the task pod obtained through watching, and determines, based on a label of each pod, a task resource on which the pod depends, to determine a condition of whether a quantity of successfully reserved resources of pods of the same task resource CR is not less than a threshold (minReservedPods) of a quantity of pods defined in the task resource.

15 **[0101]** In an embodiment, the centralized scheduler parses the task pod obtained through watching, and locates a corresponding participant based on content obtained through parsing, to obtain node resource information of the participant. The node resource information is registered in the kubernetes cluster. Therefore, the centralized scheduler allocates a proper computing node to a pod of a participant, and determines whether a quantity of pods in which a resource is reserved
20 is not less than a corresponding threshold of a quantity of pods (minReservedPods). In this way, compared with a conventional manner of querying, based on an RPC (Remote Procedure Call) or an HTTP request, whether a computing node of each participant satisfies a scheduling requirement, in this manner, scheduling efficiency can be effectively improved.

[0102] Further, if it is determined in step S29 that a quantity of pods in which a participant
25 reserves a resource is greater than or equal to the threshold of the quantity of pods in the task resource, step S30 is performed, to update the state of the task resource to the reserved state (Reserved). In an embodiment, the centralized scheduler locks a reserved resource of the participant.

[0103] It should be noted that if duration in which a participant reserves a resource
30 (resourceReservedSeconds) is defined in the task resource, the centralized scheduler can start timing from a moment at which the state of the task resource is updated to the reserved state, and

if a timing reaches the duration in which a participant reserves a resource and no pod of the task resource is scheduled (for example, it is not learned through watching that the state of the task resource is changed to "schedulable"), release the reserved resource of the participant, and change the state of the task resource to the failed state.

5 **[0104]** When it is determined in step S29 that the quantity of pods in which a participant reserves a resource is less than the threshold of the quantity of pods in the task resource, waiting is performed, until the corresponding state of the task resource group is updated to "failed" or the reserved state reaches duration in which a resource is reserved (resourceReservedSeconds), the reserved resource is released, and the state of the task resource is changed to "failed".

10 **[0105]** Step S31: The centralized scheduler determines, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

[0106] It should be noted that there are a plurality of implementations of step S31. In an implementation, the centralized scheduler determines a quantity of task resources in a reserved state in task resources in the same task resource group, that is, a quantity of participants that have
15 successfully reserved resources, and determines whether the quantity of task resources (or the quantity of participants) is not less than a threshold of a quantity of task resources corresponding to the task resource group (or a minimum quantity of participants).

[0107] Another implementation of this step is described below with reference to FIG. 3. Substeps 301 to 305 shown in FIG. 3 are specifically included.

20 **[0108]** Step 301: A task resource group controller learns, by using a watch (watch) mechanism, that a state of a task resource CR in a namespace of a participant is changed to a reserved state.

[0109] Step 302: The task resource group controller determines whether a quantity of task resources that satisfy a reserved state in a task resource CR in the same task resource group CR is greater than or equal to a corresponding threshold of a quantity of task resources. It should be
25 understood that the threshold of the quantity of task resources can be obtained from content included in the task resource group CR, and is actually a value of a parameter minReservedMembers.

[0110] Step 303: The task resource group controller updates all task resource CRs in the task resource group CR to a schedulable state (schedulable) when determining that the quantity of task
30 resources that satisfy the reserved state in the task resource CR in the same task resource group CR is greater than or equal to the corresponding threshold of the quantity of task resources.

[0111] It should be noted that, when the task resource group controller determines that the quantity of task resources that satisfy the reserved state in the task resource CR in the same task resource group CR is less than the corresponding threshold of the quantity of task resources, in other words, if a total quantity of task resource CRs in a reservable (Reserving) state and a reserved (Reserved) state is greater than or equal to minReservedMembers, waiting continues. Otherwise, the state of the task resource group CR is changed to a reserve failed state (ReserveFailed), and the state of all the task resource CRs is changed to "failed (Failed)".

[0112] Then, in an embodiment, if the task resource group CR is in the reserve failed state (ReserveFailed), the task resource group controller changes the state of the task resource group CR to a reservable state (Reserving), and changes the state of the task resource CR to a reservable state (Reserving), based on a rescheduling waiting duration (retryDurationSeconds) defined in the task resource group CR and after a specified waiting time expires, so that a next round of scheduling is performed for the privacy computing task.

[0113] In an embodiment, information about the task resource group further includes a lifecycle (lifecycleSeconds) of the task resource group. Therefore, the task resource group controller updates the state of the task resource group CR to "failed (Failed)" if it is determined that the lifecycle ends and scheduling of the privacy computing task is not completed (for example, in the lifecycle, a quantity of task resource CRs in the reserved state in the task resource group CR is always less than the corresponding threshold of the quantity of task resources). It should be understood that a failed state (Failed) is different from the reserve failed state (ReserveFailed). After the state of the task resource group CR is changed to the former, scheduling of the privacy computing task is completely completed. After the state of the task resource group CR is changed to the latter, a next round of scheduling of the privacy computing task is further enabled.

[0114] Based on this, in an embodiment, after learning through watching that the state of the task resource CR is changed to "failed", the centralized scheduler stops related scheduling. In another implementation, after detecting that the state of the task resource group CR is changed to "failed", the privacy task controller updates a task state of the privacy task CR to "failed", and/or deletes resources such as the task resource group CR, a task pod, and a service discovery (Service).

[0115] Step 304: The centralized scheduler learns, by using a watch (watch) mechanism, that a state of a task resource CR in a namespace of a participant is changed to a schedulable state (schedulable).

[0116] Step 305: The centralized scheduler completes pod scheduling in a corresponding namespace of a participant based on the schedulable state of the task resource CR.

[0117] Specifically, the centralized scheduler binds the task pod in the namespace of the participant to a specific computing node of the participant, and a resource is reserved for the task pod in the specific computing node. In addition, a corresponding quantity of pods can be scheduled based on a quantity of pods that are actually reserved by the participant. It can be understood that the quantity of pods is not less than the threshold of the quantity of pods (minReservedPods) defined in the task resource CR.

[0118] In conclusion, according to the privacy computing task execution method disclosed in this embodiment of this specification, cross-mechanism pod scheduling can be implemented in a multi-party privacy computing scenario, and efficiency and resource utilization of cross-mechanism scheduling are effectively improved. Details are as follows:

[0119] (1) A privacy task CRD and a privacy task controller are newly added, to implement management of a multi-party secure computing task. The privacy task controller watches and parses the privacy task CR, and then creates a task scheduling and running related resource such as a task resource group (TaskResourceGroup), a task pod, and a service discovery (Service).

[0120] (2) A task resource group CRD, a task resource CRD, and a task resource group controller are newly added, to manage the task resource group CR, the task resource CR, and the task lifecycle. The task resource group controller implements coordinated scheduling at a level of a participant (Parties) by using a minimum quantity of participants (minReservedMembers) defined in the task resource group CR. The task resource group controller updates the state of the task resource (TaskResource) CR in a namespace of each participant to "schedulable (Schedulable)" only when all participants satisfy a condition of the minimum quantity of participants (minReservedMembers).

[0121] (3) The centralized scheduler implemented based on a k8s-based scheduler extension plug-in completes scheduling and management of a task pod at a level of the namespace of the participant. When the task pod satisfies a condition of the minimum quantity of pods in which a participant has reserved a resource (minReservedPods), the centralized scheduler updates the state of the task resource to "reserved" (Reserved). In this case, only a resource is reserved for the pod, but the pod is not bound to a specific computing node (node). When the task resource is changed to the schedulable state (Schedulable), the centralized scheduler binds the task pod to a specific

computing node (node).

[0122] (4) Two layers of coordinated scheduling of a multi-party security computing task are implemented based on the descriptions in (2) and (3). In this solution, a quantity of participants (Parties) and a quantity of pods in a namespace of a participant are custom, and are applicable to different scenarios. In addition, two layers of coordinated scheduling are performed, to improve node resource utilization, and avoid task scheduling and task resource waste in advance due to insufficient resources of some participants.

[0123] (5) The rescheduling waiting duration (retryDurationSeconds) defined in the task resource group and the lifecycle (lifecycleSeconds) are defined in the task resource group CR, to retry a failed task and manage a maximum retry lifecycle.

[0124] (6) Duration in which a participant reserves a resource (resourceReservedSeconds) is defined in the task resource CR, so that when a task finally cannot be scheduled normally due to insufficient resources of some participants, successfully reserved resources of some participants are released in a timely manner, and the released resources are used for another task, thereby improving resource utilization and task execution efficiency.

[0125] Corresponding to the privacy computing task execution method, an embodiment of this specification further discloses a module structure of a privacy task controller, a task resource group controller, and a centralized scheduler that are used in the method.

[0126] FIG. 4 is a schematic diagram illustrating a module structure of a privacy task controller in a k8s cluster, according to an embodiment of this specification. As shown in FIG. 4, the privacy task controller 400 includes:

a task resource group creation module 410, configured to create a corresponding task resource group based on description information of a submitted privacy computing task, where information about the task resource group includes information about a plurality of participants of the privacy computing task; and a pod creation module 420, configured to create, in each of namespaces of the plurality of participants, several pods used to run the privacy computing task.

[0127] It should be noted that the kubernetes cluster further includes a task resource group controller, configured to: learn through watching that the task resource group is created, and create a task resource in the task resource group in each of the namespaces of the plurality of participants, where information about the task resource includes a threshold of a quantity of pods in which a participant needs to reserve a resource; and the kubernetes cluster further includes a centralized

scheduler, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

[0128] In an embodiment, the privacy task controller 400 further includes a resource deleting module 430, configured to delete a related task resource group and a related task pod after it is learned through watching that the state of the task resource group is "failed".

[0129] In an embodiment, the privacy task controller 400 further includes a task state updating module 440, configured to update a task state of a privacy task CR to "failed" after it is learned through watching that the state of the task resource group is "failed".

[0130] FIG. 5 is a schematic diagram illustrating a module structure of a task resource group controller in a k8s cluster, according to an embodiment of this specification. It should be noted that the kubernetes cluster further includes a privacy task controller, configured to: create a corresponding task resource group based on description information of a submitted privacy computing task, where information about the task resource group includes information about a plurality of participants of the privacy computing task; and create, in each of namespaces of the plurality of participants, several pods used to run the privacy computing task. As shown in FIG. 5, a task resource group controller 500 includes:

a task resource creation module 510, configured to: learn through watching that the task resource group is created, and create a task resource in the task resource group in each of the namespaces of the plurality of participants, where information about the task resource includes a threshold of a quantity of pods in which a participant needs to reserve a resource.

[0131] It should be noted that the kubernetes cluster further includes a centralized scheduler, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

[0132] In an embodiment, the task resource group controller 500 further includes a first state updating module 520, configured to update each task resource in the task resource group to a

schedulable state when it is determined that a quantity of task resources in a reserved state in the task resource group is not less than a corresponding threshold of a quantity of task resources.

[0133] In an embodiment, the information about the task resource group further includes a threshold of a quantity of task resources that have been in the reserved state. The task resource group controller 500 further includes: a determining module 530, configured to: when it is determined that a quantity of task resources in a reserved state in the task resource group is less than the corresponding threshold of the quantity of task resources, determine whether a quantity of task resources in a reservable state and a reserved state in the task resource group is greater than or equal to the threshold of the quantity of task resources; and a second state updating module 540, configured to: when it is determined that the quantity of task resources in the reservable state and the reserved state in the task resource group is less than the threshold of the quantity of task resources, change the task resource group to a reserve failed state, and change each task resource to a failed state.

[0134] In a specific embodiment, the information about the task resource group further includes rescheduling waiting duration. The task resource group controller 500 further includes: a rescheduling module 550, configured to: after the waiting duration expires after the task resource group is updated to the reserve failed state, change the state of the task resource group and the state of each task resource to a reservable state, to enable a next round of scheduling for the privacy computing task.

[0135] In an embodiment, the information about the task resource group further includes a lifecycle of the task resource group. The task resource group controller 500 further includes a third state updating module 560, configured to update the state of the task resource group to "failed" when it is determined that the lifecycle ends and scheduling of the privacy computing task is not completed.

[0136] FIG. 6 is a schematic diagram illustrating a module structure of a centralized scheduler in a k8s cluster, according to an embodiment of this specification. It should be noted that the kubernetes cluster further includes a privacy task controller, configured to: create a corresponding task resource group based on description information of a submitted privacy computing task, where information about the task resource group includes information about a plurality of participants of the privacy computing task; and create, in each of namespaces of the plurality of participants, several pods used to run the privacy computing task. The kubernetes cluster further

includes a task resource group controller, configured to: learn through watching that the task resource group is created, and create a task resource in the task resource group in each of the namespaces of the plurality of participants, where information about the task resource includes a threshold of a quantity of pods in which a participant needs to reserve a resource.

5 **[0137]** As shown in FIG. 6, a centralized scheduler 600 includes:
 a state updating module 610, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and a scheduling
10 module 620, configured to determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

[0138] In an embodiment, the scheduling module 620 is specifically configured to: learn through watching that the state of the task resource is updated to a schedulable state, and when a quantity of pods in which a resource is reserved in the task resource is not less than the
15 corresponding threshold of the quantity of pods, schedule the pod to a computing node of a corresponding participant.

[0139] In an embodiment, the information about the task resource further includes duration in which a participant reserves a resource. The centralized scheduler 600 further includes a resource release module 630, configured to: if it is determined that a timing reaches the duration and it is
20 not learned through watching that the reserved state of the task resource is updated to the schedulable state, release a reserved resource of a corresponding participant, and/or change the state of the task resource to "failed".

[0140] An embodiment of another aspect further provides a computer-readable storage medium. The computer-readable storage medium stores a computer program, and when the
25 computer program is executed on a computer, the computer is enabled to perform the method described with reference to FIG. 2 or FIG. 3.

[0141] An embodiment of still another aspect further provides a computing device, including a memory and a processor. The memory stores executable code, and when the processor executes the executable code, the method described with reference to FIG. 2 or FIG. 3 is implemented.

30 **[0142]** A person skilled in the art should be aware that in the one or more examples, functions described in the present invention may be implemented by hardware, software, firmware, or any combination thereof. When the functions are implemented by using software, the functions may

be stored in a computer-readable medium or transmitted as one or more instructions or code in the computer-readable medium.

[0143] In the above-mentioned specific implementations, the objectives, technical solutions, and beneficial effects of the present invention are further described in detail. It should be
5 understood that the above-mentioned descriptions are merely specific implementations of the present invention, but are not intended to limit the protection scope of the present invention. Any modification, equivalent replacement, improvement, or the like made based on the technical solutions of the present invention shall fall within the protection scope of the present invention.

CLAIMS

What is claimed is:

1. A privacy computing task execution method, related to a privacy task controller, a task resource group controller, and a centralized scheduler in a kubernetes cluster, wherein the method
5 comprises:

creating, by the privacy task controller, a corresponding task resource group based on description information of a submitted privacy computing task, wherein information about the task resource group comprises information about a plurality of participants of the privacy computing task;

10 creating, by the privacy task controller in each of namespaces of the plurality of participants, several pods used to run the privacy computing task;

learning, by the task resource group controller through watching, that the task resource group is created, and creating a task resource in the task resource group in each of the namespaces of the plurality of participants, wherein information about the task resource comprises a threshold of a
15 quantity of pods in which a participant needs to reserve a resource;

learning, by the centralized scheduler through watching, that any of the several pods is created, and updating a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and

20 determining, by the centralized scheduler based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

2. The method according to claim 1, wherein the information about the task resource group further comprises a threshold of a quantity of task resources that have been in the reserved state, and the method further comprises:

25 updating, by the task resource group controller, each task resource in the task resource group to a schedulable state if the task resource group controller determines that the quantity of task resources that have been in the reserved state in the task resource group is not less than the corresponding threshold of the quantity of task resources.

3. The method according to claim 2, wherein the determining, by the centralized scheduler
30 based on the state of the task resource, whether to schedule the pod to a computing node of a

corresponding participant comprises:

learning, by the centralized scheduler through watching, that the state of the task resource is updated to the schedulable state, and scheduling the pod to the computing node of the corresponding participant if the quantity of pods in which a resource is reserved in the task resource
5 is not less than the corresponding threshold of the quantity of task resources.

4. The method according to claim 2, wherein the information about the task resource further comprises duration in which a participant reserves a resource, and after the state of the task resource is updated to the reserved state, the method further comprises:

if the centralized scheduler determines that a timing reaches the duration and does not learn
10 through watching that the reserved state of the task resource is updated to the schedulable state, releasing, by the centralized scheduler, a reserved resource of a corresponding participant, and/or changing the state of the task resource to "failed".

5. The method according to claim 1, wherein the information about the task resource group further comprises a threshold of a quantity of task resources that have been in the reserved state,
15 and the method further comprises:

if the task resource group controller determines that the quantity of task resources in the reserved state in the task resource group is less than the corresponding threshold of the quantity of task resources, determining, by the task resource group controller, whether a quantity of task resources in a reservable state and the reserved state in the task resource group is greater than or
20 equal to the threshold of the quantity of task resources;

continuing waiting if it is determined that the quantity of task resources in the reservable state and the reserved state in the task resource group is greater than or equal to the threshold of the quantity of task resources; and

if it is determined that the quantity of task resources in the reservable state and the reserved
25 state in the task resource group is less than the threshold of the quantity of task resources, changing the task resource group to a reserve failed state, and changing each task resource in the task resource group to a failed state.

6. The method according to claim 5, wherein the information about the task resource group further comprises rescheduling waiting duration; and after the changing the task resource group to a reserve failed state, and changing each task resource in the task resource group to a failed state,
30 the method further comprises:

after the waiting duration expires, separately changing, by the task resource group controller, states of the task resource group and each task resource to the reservable state, to enable a next round of scheduling for the privacy computing task.

7. The method according to any one of claims 2 to 6, wherein the information about the task resource group further comprises a lifecycle of the task resource group; and the method further comprises:

updating, by the task resource group controller, the state of the task resource group to "failed" when the task resource group controller determines that the lifecycle ends and scheduling of the privacy computing task is not completed.

8. The method according to claim 7, wherein after the updating the state of the task resource group to "failed", the method further comprises:

after the privacy task controller learns through watching that the state of the task resource group is "failed", updating, by the privacy task controller, a task state of the privacy computing task to "failed", and/or deleting the task resource group and the several pods.

9. The method according to any one of claims 1 to 6 or 8, wherein the description information of the privacy computing task comprises at least one of the following: the information about the plurality of participants, the threshold of the quantity of pods in which a participant needs to reserve a resource, or the threshold of the quantity of task resources that have been in the reserved state.

10. A privacy task controller in a kubernetes cluster, comprising:

a task resource group creation module, configured to create a corresponding task resource group based on description information of a submitted privacy computing task, wherein information about the task resource group comprises information about a plurality of participants of the privacy computing task; and

a pod creation module, configured to create, in each of namespaces of the plurality of participants, several pods used to run the privacy computing task, wherein

the kubernetes cluster further comprises a task resource group controller, configured to: learn through watching that the task resource group is created, and create a task resource in the task resource group in each of the namespaces of the plurality of participants, wherein information about the task resource comprises a threshold of a quantity of pods in which a participant needs to reserve a resource; and

the kubernetes cluster further comprises a centralized scheduler, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and
5 determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

11. A task resource group controller in a kubernetes cluster, wherein the kubernetes cluster further comprises a privacy task controller, configured to: create a corresponding task resource group based on description information of a submitted privacy computing task, wherein
10 information about the task resource group comprises information about a plurality of participants of the privacy computing task; and create, in each of namespaces of the plurality of participants, several pods used to run the privacy computing task; and the task resource group controller comprises:

a task resource creation module, configured to: learn through watching that the task resource
15 group is created, and create a task resource in the task resource group in each of the namespaces of the plurality of participants, wherein information about the task resource comprises a threshold of a quantity of pods in which a participant needs to reserve a resource, wherein

the kubernetes cluster further comprises a centralized scheduler, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved
20 state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

12. A centralized scheduler in a kubernetes cluster, wherein the kubernetes cluster further
25 comprises a privacy task controller, configured to: create a corresponding task resource group based on description information of a submitted privacy computing task, wherein information about the task resource group comprises information about a plurality of participants of the privacy computing task; and create, in each of namespaces of the plurality of participants, several pods used to run the privacy computing task;

30 the kubernetes cluster further comprises a task resource group controller, configured to: learn through watching that the task resource group is created, and create a task resource in the task

resource group in each of the namespaces of the plurality of participants, wherein information about the task resource comprises a threshold of a quantity of pods in which a participant needs to reserve a resource; and

the centralized scheduler comprises:

5 a state updating module, configured to: learn through watching that any of the several pods is created, and update a state of the task resource to a reserved state when it is determined that a quantity of pods in which a resource is reserved in a task resource on which the pod depends is not less than the corresponding threshold of the quantity of pods; and

10 a scheduling module, configured to determine, based on the state of the task resource, whether to schedule the pod to a computing node of a corresponding participant.

13. A computer-readable storage medium, wherein the computer-readable storage medium stores a computer program, and when the computer program is executed on a computer, the computer is enabled to perform the method according to any one claims 1 to 9.

14. A computing device, comprising a memory and a processor, wherein the memory stores
15 executable code, and when the processor executes the executable code, the method according to any one of claims 1 to 9 is implemented.