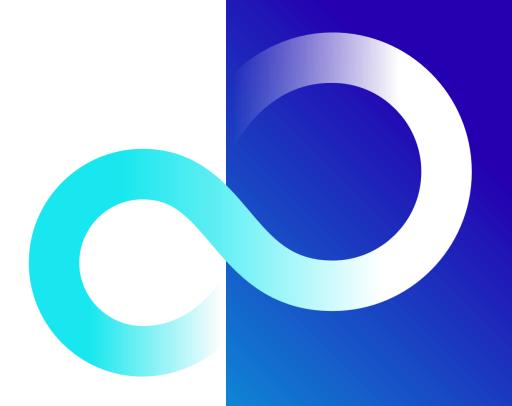


Oracle Solaris 11 CPUリソース制限 設定手順書



2019年9月

第1.1版

富士通株式会社



## ■使用条件

 著作権・商標権・その他の知的財産権について コンテンツ(文書・画像・音声等)は、著作権・商標権・その他の知的財産権で保護されています。
 本コンテンツは、個人的に使用する範囲でプリントアウトまたはダウンロードできます。ただし、これ以外の利用(ご自分のページへの再利用や他のサーバへのアップロード等)については、当社または権利者の許諾が必要となります。

• 保証の制限

本コンテンツについて、当社は、その正確性、商品性、ご利用目的への適合性等に関して保証するものではなく、そのご利用により生じた損害について、当社は法律上のいかなる責任も負いかねます。本コンテンツは、予告なく変更・廃止されることがあります。

## ■商標について

- UNIX は、米国およびその他の国におけるオープン・グループの登録商標です。
- SPARC Enterprise、SPARC64、SPARC64 ロゴ、およびすべての SPARC 商標は、米国 SPARC International, Inc.
   のライセンスを受けて使用している、同社の米国およびその他の国における商標または登録商標です。
- Oracle と Java は、Oracle Corporation およびその子会社、関連会社の米国およびその他の国における登録商標です。
- その他各種製品名は、各社の製品名称、商標または登録商標です。



# はじめに

## 本書の内容

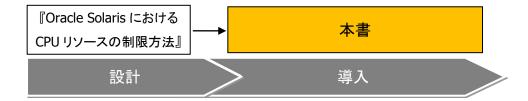
本書は、Oracle Solaris 11 環境のアプリケーションが利用する CPU リソースを制限する手順について記載しています。

## 留意事項

- 本書では Oracle Solaris を Solaris と記載することがあります。
- 本書では Oracle Solaris ゾーンをゾーン、non-global zone と記載することがあります。

## ドキュメントの位置付け

• 本書は以下の位置づけになります。





## 目 次

1. Oracle Solaris 環境での CPU リソース制限方式	1
1-1. CPU リソースの制限方法	1
1-2. CPU リソース制限の設定対象	1
2. CPU リソース制限設定手順	2
2-1. リソースプール方式	2
2-1-1. リソースプールの作成	3
2-1-2. プロジェクトの設定(アプリケーションの実行が global zone の root 権限の場合)	4
2-1-3. プロジェクトの作成と設定(アプリケーションの実行が global zone の一般ユーザ-	-の場合)5
2-1-4. ゾーンの設定(アプリケーションの実行環境が non-global zone の場合)	6
2-2. CPU キャップ方式	7
2-2-1. プロジェクトの設定(アプリケーションの実行が global zone の root の場合)	8
2-2-2. プロジェクトの作成と設定(アプリケーションの実行が global zone の一般ユーザ-	一の場合)
	10
2-2-3. ゾーンの設定(アプリケーションの実行環境が non-global zone の場合)	12
3. CPU リソース制限確認手順	13
3-1. リソースプール方式	13
3-1-1. リソースプールの確認	13
3-1-2. ゾーンのリソースプール設定/状態の確認	17
3-2. CPU キャップ方式	19
3-2-1. プロジェクトの CPU キャップ設定/状態の確認	19
3-2-2. ゾーンの CPU キャップ設定/状態の確認	20
4.《参考》CPU リソース制限設定の削除手順	22
4-1. CPU リソース制限パラメーターの削除	22
4-1-1. プロジェクトの場合の設定値の削除	22
4-1-2. ゾーンの場合の設定値の削除	23
4-2 リソースプール構成の削除	24



## 1. Oracle Solaris 環境での CPU リソース制限方式

#### 1-1.CPU リソースの制限方法

Solaris 環境上のアプリケーションが利用する CPU リソースを制限する方法には、リソースプールを利用する「リソースプール方式」と、CPU 利用率の上限を設定する「CPU キャップ方式」の 2 つがあります。

#### 1) リソースプール方式

- ・CPU をコア、スレッド単位でまとめたリソースプールを構成し、アプリケーションが利用するリソースプールとして指定する方式です。
- ・ゾーンまたはプロジェクトは一つのリソースプールに結び付けることが可能です。

#### 2) CPU キャップ方式

- ・アプリケーションから利用可能な CPU リソースに対して、利用率の上限値(CPU キャップ)を設定する方式です。1%単位で設定することが可能です。
- ・CPU 利用率の上限値はゾーンまたはプロジェクトのパラメーターとして指定可能です。

## 1-2.CPU リソース制限の設定対象

Solaris 環境上のアプリケーションの動作環境によって、ゾーンまたはプロジェクトに対して CPU リソースの制限を設定します。

#### 1) Oracle Solaris ゾーン

- ・non-global zone で動作するアプリケーションについては、global zone 上で non-global zone のリソース制限を行います。
- ·zonecfg コマンドを利用してリソース制限に関するパラメーターを設定します。

### 2) プロジェクト

- ・global zone で動作するアプリケーションについては、global zone 上のプロジェクトを利用してリソース 制限を行います。
- •projadd コマンド等を利用してリソース制限に関するパラメーターを設定します。
- ・アプリケーションが特定のユーザー権限で実行される場合(oracle ユーザーなど)は、そのユーザー用のプロジェクトの作成が必要です。



# 2. CPU リソース制限設定手順

## 2-1.リソースプール方式

リソースプール方式は最初にアプリケーションが利用する専用のリソースプールを作成します。そして、アプリケーションが global zone 環境で動作する場合はプロジェクト、non-global zone 環境で動作する場合は non-global zone を作成し、作成したリソースプールを指定します。

ここでは、以下の環境での動作を想定したアプリケーション専用のリソースプールを作成します。

リソースプール名	プロセッサセット名	最小 CPU 数	最大 CPU 数	備考
pool_default	pset_default	1	65545	global zone 専用
pool_1	pset_1	8	8	アプリケーション用

<sup>※</sup>指定する CPU の単位は、CPU スレッドです。

アプリケーションの動作環境が global zone の場合は、アプリケーションを実行するユーザーのプロジェクトの定義情報にソースプールを指定します。アプリケーションを実行するユーザー(権限)が root 以外の場合は、ユーザー用のプロジェクトを新規作成します。

アプリケーションの動作環境が non-global zone の場合は、non-global zone の定義情報にリソースプールを指定します。non-global zone の場合、アプリケーションを実行するユーザーを考慮する必要はありません。

ここでは、以下のパターンを想定して定義します。

No.	動作環境	実行ユーザー(権限)	設定対象		リソースプール
1	global zone	root	プロジェクト	user.root	pool_1
2	global zone	一般ユーザー	プロジェクト	user.user01	pool_1
		(user01)			
3	non-global zone	-(任意)	ゾーン	zone01	pool_1

<sup>※1</sup>コアあたりのスレッド数はハードウェアの機種によって異なります。



#### 2-1-1.リソースプールの作成

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) リソースプールのサービスの状態を確認します。

# svcs svc:/system/pools:default STATE STIME FMRI

disabled May\_28 svc:/system/pools:default

- ▼ 初期状態ではリソースプールサービスは、無効(disabled)です。
- 3) リソースプールのサービスを起動します。

# svcadm enable svc:/system/pools:default

4) リソースプール構成を確認します。

# poolstat -r all id pool type rid rset

min max size used load 1 66K 16 0.00 0.01

- 初期状態では pool\_default のみです。
- 5) 現在のリソースプール構成をファイルに保存します。

#### # pooladm -s

0 pool\_default

● /etc/pooladm.conf ファイルが新規作成されます。ファイルが存在する場合は、上書きされます。

pset -1 pset\_default

- 6) アプリケーション用のリソースプール(pool\_1)を新規作成します。
- # poolcfg -c 'create pool pool\_1'
- # poolcfg -c 'create pset pset\_1(uint pset.min=8;uint pset.max=8)'
- # poolcfg -c 'associate pool pool 1(pset pset 1)'
- pset.min 値に最小 CPU 数、pset.max 値に最大 CPU 数を指定します。
- 7) リソースプール構成を反映します。
- # pooladm -c
- 8) リソースプール構成を確認します。

# poolstat -r all
id pool type rid rset min max size used load
0 pool\_default pset -1 pset\_default 1 66K 8 0.00 0.01
1 pool\_1 pset 1 pset\_1 8 8 8 0.00 0.00

■ pool\_1 が表示されます。size の値が各リソースプールに割り当てられている CPU 数を表します。



#### 2-1-2.プロジェクトの設定(アプリケーションの実行が global zone の root 権限の場合)

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) root のプロジェクト(user.root) 定義を確認します。

```
# projects -l user.root
user.root
projid : 1
comment: ""
users : (none)
groups : (none)
attribs:
```

- 初期状態では何も定義されていません。
- 3) root のプロジェクト(user.root)定義にリソースプール(pool\_1)を指定します。

```
# projmod -K 'project.pool=pool_1' user.root
# projects -l user.root
user.root
    projid : 1
    comment: ""
    users : (none)
    groups : (none)
    attribs: project.pool=pool_1
```

- attribs に project.pool パラメーターが表示されます。
- 4) OS を再起動します。起動後、一般ユーザーでログイン後、root の役割を引き受けます。

```
# shutdown -y -g0 -i6
```

5) root のプロジェクトを確認します。

```
# id -p
uid=0(root) gid=0(root) projid=1(user.root)
```

- user.rootプロジェクトに属している事を確認します。
- 6) root のリソースプールを確認します。

```
# poolbind -q $$
27597 pool_1
```

- \$\$を指定することでログインシェルのプロセスを確認します。
- 7) アプリケーションが利用するリソースプールを確認します。

#### # poolbind -q [プロセス ID]

- ▼ アプリケーションのプロセス ID は、事前に ps コマンド等で調べます。
- OS 起動後に root 権限で起動したアプリケーション(プロセス)に対してリソースプール名(pool\_1)が割り当てられます。



## 2-1-3.プロジェクトの作成と設定(アプリケーションの実行が global zone の一般ユーザーの 場合)

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) user01 のプロジェクト(user.user01)を新規作成し、リソースプール(pool\_1)を指定します。

#### # projadd -K 'project.pool=pool 1' user.user01

- user01 ユーザーは事前に作成しておいてください。
- 3) user01 のプロジェクト(user.user01) 定義を確認します。

# projects -l user.user01 user.user01

projid: 100 comment: "" users: (none) groups: (none)

attribs: project.pool=pool\_1

- attribs に project.pool パラメーターが表示されます。
- 4) OS 再起動後、user01 ユーザーでログインします。

#### # shutdown -y -g0 -i6

5) user01 ユーザーのプロジェクトを確認します。

## \$ id -p uid=1000(user01) gid=1000(group01) projid=100(user.user01)

- user.user01 プロジェクトに属している事を確認します。
- 6) user01 ユーザーのリソースプールを確認します。

# \$ poolbind -q \$\$ 1249 pool\_1

- \$\$を指定することでログインシェルのプロセスを確認します。
- 7) アプリケーションが利用するリソースプールを確認します。

## \$ poolbind -q [プロセス ID] [プロセス ID] pool\_1

- ▼ アプリケーションのプロセス ID は、事前に ps コマンド等で調べます。
- ▼ アプリケーション用のリソースプール名(pool\_1)が表示されることを確認します。



## 2-1-4. **ゾーンの**設定(アプリケーションの実行環境が non-global zone の場合)

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) non-global zone の構成定義を実行します。

# zonecfg -z zone01 zonecfg:zone01> set pool=pool 1 zonecfg:zone01> export set zonepath=/export/zones/zone01 set autoboot=false

← ★リソースプールの指定

set ip-type=exclusive add anet set linkname=net0 set lower-link=auto

set pool=pool\_1

set configure-allowed-address=false

set link-protection=mac-nospoof

set mac-address=random

end

zonecfg:zone01> exit

- 『プメーター(set pool=pool 1)が定義されていることを確認します。
- non-global zone はインストール済みであることとします。その他のパラメーターは任意です。
- 3) non-global zone を起動(起動済みの場合は再起動)し、zone にログインします。

# zoneadm -z zone01 boot # zlogin zone01

4) non-global zone を起動(起動済みの場合は再起動)し、zone にログインします。

zone01# poolbind -q \$\$ 2498 pool\_1

● \$\$を指定することでログインシェルのプロセスを確認します。



## 2-2.CPU キャップ方式

CPU キャップ方式は、アプリケーションが動作する環境(global zone または non-global zone)に応じて、cpu-cap パラメーターに CPU キャップ値を設定します。アプリケーションが global zone 環境で動作する場合はプロジェクト、non-global zone 環境で動作する場合は non-global zone を作成して、cpu-cap パラメーターに CPU キャップ値を指定します。

ここでは、以下のパターンを想定しアプリケーション専用の CPU キャップを設定します。

No.	動作環境	実行ユーザー(権限)	設定対象		CPU キャップ値
1	global zone	root	プロジェクト	user.root	800
2	global zone	一般ユーザー	プロジェクト	user.user01	800
		(user01)			
3	non-global zone	-(任意)	ゾーン	zone01	800

<sup>※</sup>指定する CPU キャップ値 100=1 スレッドです。

アプリケーションの動作環境が global zone の場合は、アプリケーションを実行するユーザーのプロジェクトの cpu-cap パラメーターに CPU キャップ値を指定します。アプリケーションを実行するユーザー(権限)が root 以外の場合は、ユーザー用のプロジェクトを新規作成して、cpu-cap パラメーターに CPU キャップ値を指定します。

アプリケーションの動作環境が non-global zone の場合は、non-global zone の cpu-cap パラメーターに CPU キャップ値を指定します。non-global zone の場合、アプリケーションを実行するユーザーを考慮する 必要はありません。

<sup>※1</sup>コアあたりのスレッド数はハードウェアの機種によって異なります。



#### 2-2-1.プロジェクトの設定(アプリケーションの実行が global zone の root の場合)

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) root のプロジェクト(user.root)定義を確認します。

```
# projects -l user.root
user.root
projid : 1
comment: ""
users : (none)
groups : (none)
attribs:
```

- 初期状態では何も定義されていません。
- 3) root のプロジェクト(user.root)定義に CPU キャップ値(800)を指定します。

```
# projmod -K 'project.cpu-cap=(privileged,800,deny)' user.root
# projects -l user.root
user.root
projid : 1
comment: ""
users : (none)
groups : (none)
attribs: project.cpu-cap=(privileged,800,deny)
```

- attribs に project.cpu-cap パラメーターが表示されます。
- 4) OS を再起動します。起動後、一般ユーザーでログイン後、root の役割を引き受けます。

```
# shutdown -y -g0 -i6
```

5) root のプロジェクトを確認します。

```
# id -p
uid=0(root) gid=0(root) projid=1(user.root)
```

- user.root プロジェクトに属している事を確認します。
- 6) root の CPU キャップ値を確認します。

```
# prctl -n project.cpu-cap $$
process: 2522: -bash
NAME
         PRIVILEGE
                                   FLAG
                                           ACTION
                                                             RECIPIENT
                          VALUE
project.cpu-cap
       usage
                              1
       privileged
                           800
                                           deny
                          4.29G
       system
                                    inf
                                           deny
```

- \$\$を指定することでログインシェルのプロセスを確認します。
- privileged 権限の VALUE 値が設定値となります。



## 7) アプリケーションの CPU キャップ値を確認します。

	n project.cpu-ca [プロセス ID]: [ フ		D]		
NAME	PRIVILEGE	<b>VALUE</b>	FLAG	<b>ACTION</b>	RECIPIENT
project.c	pu-cap				
u	ısage	1			
p	rivileged	800	-	deny	-
s	ystem	4.29G	inf	deny	-

- ▼ アプリケーションのプロセス ID は、事前に ps コマンド等で調べます。
- ▼ アプリケーション用の CPU キャップ値が 800 であることを確認します。
- OS 起動後に root 権限で起動したアプリケーション(プロセス)に対して CPU キャップ値が有効になります。



# 2-2-2.プロジェクトの作成と設定(アプリケーションの実行が global zone の一般ユーザーの 場合)

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) user01 のプロジェクト(user.user01)を新規作成し、CPU キャップ値(800)を指定します。

### # projadd -K 'project.cpu-cap=(privileged,800,deny)' user.user01

- user01 ユーザーは事前に作成しておいてください。
- 3) user01 のプロジェクト(user.user01) 定義を確認します。

# projects -l user.user01
user.user01
projid : 100
comment: ""
users : (none)
groups : (none)

attribs: project.cpu-cap=(privileged,800,deny)

- attribs に project.cpu-cap パラメーターが表示されます。
- 4) OS 再起動後、user01 ユーザーでログインします。

#### # shutdown -y -g0 -i6

5) user01 ユーザーのプロジェクトを確認します。

## \$ id -p uid=1000(user01) gid=1000(group01) projid=100(user.user01)

- user.user01 プロジェクトに属している事を確認します。
- 6) user01 ユーザーの CPU キャップ値を確認します。

	project.cpu-ca 2545: -bash	р \$\$			
NAME	PRIVILEGE	VALUE	FLAG	ACTION	RECIPIENT
project.c	pu-cap				
u	sage	1			
p	rivileged	800	=	deny	-
S	ystem	4.29G	inf	deny	-

- \$\$を指定することでログインシェルのプロセスを確認します。
- privileged 権限の VALUE 値が設定値となります。



## 7) アプリケーションの CPU キャップ値を確認します。

	-n project.cpu-c : [プロセス ID]: [		D]		
NAME	<b>PRIVILEGE</b>	VALUE	FLAG	<b>ACTION</b>	RECIPIENT
project.	сри-сар				
	usage	1			
	privileged	800	-	deny	-
	system	4.29G	inf	deny	-

- ▼ アプリケーションのプロセス ID は、事前に ps コマンド等で調べます。
- ▼ アプリケーションのプロセスの CPU キャップ値が 800 であることを確認します。



#### 2-2-3.ゾーンの設定(アプリケーションの実行環境が non-global zone の場合)

1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。 non-global zone の構成定義を実行します。

# zonecfg -z zone01
zonecfg:zone01> add capped-cpu
zonecfg:zone01:capped-cpu> set ncpus=8
zonecfg:zone01:capped-cpu> end

■ ncpus パラメーターには 1 スレッド=1 で指定します。

2) non-global zone の構成定義を確認します。

zonecfg:zone01> export create -b set zonepath=/export/zones/zone01 set brand=solaris set autoboot=false ← ★リソースプールの指定 set pool=pool\_1 set ip-type=exclusive add anet set linkname=net0 set lower-link=auto set configure-allowed-address=false set link-protection=mac-nospoof set mac-address=random end add rctl ← ★CPU キャップ値に指定 set name=zone.cpu-cap add value (priv=privileged,limit=800,action=deny) end

- 『アラメーター(set name=zone.cpu-cap)が定義されていることを確認します。
- リソースプールの指定も定義されている場合は、そのリソースプールで指定した範囲内で CPU キャップ値が有効になります。
- 3) non-global zone を起動(起動済みの場合は再起動)します。

#### # zoneadm -z zone01 boot

4) non-global zone の CPU キャップ値を確認します。

	-n zone.cpu-cap : zone01	-i zone zone(	)1		
NAME	PRIVILEGE	VALUE	FLAG	ACTION	RECIPIENT
zone.cp	u-cap				
	usage	0			
	privileged	800	-	deny	-
	system	4.29G	inf	deny	-

rivileged 権限の VALUE 値が設定値となります。



## 3. CPU リソース制限確認手順

#### 3-1.リソースプール方式

## 3-1-1.リソースプールの確認

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) リソースプール実行中の CPU 数を確認する場合は、下記コマンドを実行します。

3) リソースプールの CPU ID を含めて確認する場合は、下記コマンドを実行します。

## # pooladm

## system default

string system.comment
int system.version 1
boolean system.bind-default true
string system.poold.objectives wt-load

#### pool pool\_1

int pool.sys\_id 1
boolean pool.active true
boolean pool.default false
int pool.importance 1
string pool.comment
pset pset\_1

#### pool pool\_default

int pool.sys\_id 0
boolean pool.active true
boolean pool.default true
int pool.importance 1
string pool.comment
pset pset default

## pset pset\_1

int pset.sys\_id 1
boolean pset.default false
uint pset.min 8
uint pset.max 8
string pset.units population
uint pset.load 0

新規作成したリソースプール (pool\_1)と結合しているプロセッ サセット(pset\_1)を表示。

デフォルトのリソースプール (pool\_default)と結合しているプロ セッサセット(pset\_default)を表示。

プロセッサセット(pset\_1)に定義 された最低 CPU 数 (pset.min)、最大 CPU 数 (pset.max)、現在の CPU 数 (pset.size)を表示。

min「最低 CPU 数」、max「最大 CPU 数」、size「現在の CPU 数」を表します。



uint pset.size 8 string pset.comment

cpu

int cpu.sys\_id 5
string cpu.comment
string cpu.status on-line

cpu

int cpu.sys\_id 4
string cpu.comment
string cpu.status on-line

cpu

int cpu.sys\_id 7
string cpu.comment
string cpu.status on-line

cpu

int cpu.sys\_id 6
string cpu.comment
string cpu.status on-line

cpu

int cpu.sys\_id 1
string cpu.comment
string cpu.status on-line

cpu

int cpu.sys\_id 0
string cpu.comment
string cpu.status on-line

cpu

int cpu.sys\_id 3
string cpu.comment
string cpu.status on-line

cpu

int cpu.sys\_id 2
string cpu.comment
string cpu.status on-line

現在の CPU の CPU ID を表示。 8core/1CPU の場合、CPU ID は 0~7がコア単位の CPU となる。

pset pset\_default

int pset.sys\_id -1 boolean pset.default true

uint pset.min 1
uint pset.max 65536
string pset.units population

uint pset.load 3
uint pset.size 8

プロセッサセット (pset\_default) に 定義された最低 CPU 数 (pset.min)、 最大 CPU 数 (pset.max)、現在の CPU 数 (pset.size) を表示。



stri	ng pset.co	omment		
сри	int string	cpu.sys_id 13 cpu.comment cpu.status on-line		
сри	int string	cpu.sys_id 12 cpu.comment cpu.status on-line		
cpu	int string	cpu.sys_id 15 cpu.comment cpu.status on-line		
cpu	int string	cpu.sys_id 14 cpu.comment cpu.status on-line	<b>\</b>	現在の CPU の CPU ID を表示。8core/1CPU の場合、CPU ID は 9~15 がコア単位の
cpu	int string	cpu.sys_id 9 cpu.comment cpu.status on-line		でPU となる。
cpu	int string	cpu.sys_id 8 cpu.comment cpu.status on-line		
cpu	int string	cpu.sys_id 11 cpu.comment cpu.status on-line		
сри	int string	cpu.sys_id 10 cpu.comment cpu.status on-line		

■ コア単位のリソースプールを構成する場合は、CPU ID の確認が必要となります。



4) プロジェクトのリソースプール設定を確認する場合は、下記コマンドを実行します。

```
# projects -I
system
       projid: 0
       comment: ""
       users : (none)
       groups: (none)
       attribs:
user.root
       projid: 1
       comment: ""
       users : (none)
       groups: (none)
       attribs:
noproject
       projid: 2
       comment: ""
       users : (none)
       groups: (none)
       attribs:
default
       projid: 3
       comment: ""
       users : (none)
       groups: (none)
       attribs:
group.staff
       projid: 10
       comment: ""
       users : (none)
       groups: (none)
       attribs:
user.user01
       projid: 100
       comment: ""
       users : (none)
       groups: (none)
       attribs: project.pool=pool_1
```

新規作成した user.user01 プロジェクトの attribs のパラメーター(project.pool)を確認します。



5) 実行中のプロジェクトが利用しているリソースプールを確認する場合は、下記コマンドを実行します。

```
# su - user01
Oracle Corporation SunOS 5.11 11.0 November 2011

$ id -p
uid=1000(user01) gid=1000(group01) projid=100(user.user01)

$ poolbind -q $$
24699 pool_1
```

- 確認したいプロジェクトに所属するユーザーに変更後、poolbind コマンドを実行して確認します。
- \$\$は現在のログインプロセスを表しますが、プロセス ID を指定して確認することも可能です。

## 3-1-2.ゾーンのリソースプール設定/状態の確認

1) ゾーンの設定を確認する場合は、下記コマンドを実行します。

```
# zonecfg -z zone01 info
zonename: zone01
zonepath: /export/zones/zone01
brand: solaris
autoboot: false
bootargs:
file-mac-profile:
pool: pool 1
limitpriv:
scheduling-class:
ip-type: exclusive
hostid:
fs-allowed:
anet:
        linkname: net0
        lower-link: auto
        allowed-address not specified
        configure-allowed-address: true
        defrouter not specified
        allowed-dhcp-cids not specified
        link-protection: mac-nospoof
        mac-address: random
        mac-prefix not specified
        mac-slot not specified
        vlan-id not specified
        priority not specified
        rxrings not specified
        txrings not specified
        mtu not specified
        maxbw not specified
        rxfanout not specified
```

<sup>■</sup> pool パラメーターを確認します。



2) 起動中のゾーンが利用しているリソースプールを確認する場合は、下記コマンドを実行します。

# zlogin zone01

[Connected to zone 'zone01' pts/2]

Oracle Corporation SunOS 5.11 11.0 November 2011

# poolbind -q \$\$ 24724 pool\_1

■ 確認したいゾーンにログイン後、poolbind コマンドを実行して確認します。



## 3-2.CPU キャップ方式

#### 3-2-1.プロジェクトの CPU キャップ設定/状態の確認

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) プロジェクトの CPU キャップ設定を確認する場合は、下記コマンドを実行します。

```
# projects -l
system
        projid: 0
        comment: ""
        users : (none)
        groups: (none)
        attribs:
user.root
        projid: 1
        comment: ""
        users : (none)
        groups: (none)
        attribs:
noproject
        projid: 2
        comment: ""
        users : (none)
        groups: (none)
        attribs:
default
        projid: 3
        comment: ""
        users : (none)
        groups: (none)
       attribs:
group.staff
        projid: 10
        comment: ""
        users : (none)
        groups: (none)
        attribs:
user.user01
        projid: 100
        comment: ""
        users : (none)
        groups: (none)
        attribs: project.cpu-cap=(privileged,800,deny)
```

- 新規作成した user.user01 プロジェクトの attribs のパラメーター(project.cpu-cap)を確認します。
- カッコ内の数値部分が設定値となり、1CPU=100 として設定されます。



3) 実行中のプロジェクトの CPU キャップ値を確認する場合は、下記コマンドを実行します。

```
# su - user01
Oracle Corporation
                      SunOS 5.11
                                      11.0
                                              November 2011
$ id -p
uid=1000(user01) gid=1000(group01) projid=100(user.user01)
$ prctl -n project.cpu-cap $$
process: 24736: -bash
NAME
       PRIVILEGE
                     VALUE
                               FLAG
                                     ACTION
                                                               RECIPIENT
project.cpu-cap
        usage
                          1
                       400
        privileged
                                      deny
        system
                     4.29G
                                inf
                                      deny
```

## 3-2-2.ゾーンの CPU キャップ設定/状態の確認

1) ゾーンの CPU キャップ設定を確認する場合は、下記コマンドを実行します。

```
# zonecfg -z zone01 info
zonename: zone01
zonepath: /export/zones/zone01
brand: solaris
autoboot: false
bootargs:
file-mac-profile:
pool: pool_1
limitpriv:
scheduling-class:
ip-type: exclusive
hostid:
fs-allowed:
anet:
        linkname: net0
       lower-link: auto
           (省略)
capped-cpu:
        [ncpus: 8.00]
rctl:
        name: zone.cpu-cap
        value: (priv=privileged,limit=800,action=deny)
```

<sup>■</sup> privileged の VALUE の値が CPU キャップ値となり、1CPU=100 として設定されます。

**c** capped-cpu の ncpus の値は 1CPU=1 として少数第二位まで表示されます。 capped-cpu のパラメーターは、rctl の zone.cpu-cap の limit の値として、1CPU=100 に変換されて表示されます。



2) ゾーン起動中の CPU キャップ値を確認する場合は、下記コマンドを実行します。

# zlogin zone01 [Connected to zone 'zone01' pts/2] **Oracle Corporation November 2011 SunOS 5.11** 11.0 # prctl -n zone.cpu-cap \$\$ process: 24756: -bash NAME PRIVILEGE VALUE FLAG ACTION **RECIPIENT** zone.cpu-cap usage 0 800 privileged deny system 4.29G inf deny

privileged の VALUE の値が CPU キャップ値となり、1CPU=100 として設定されます。



## 4.《参考》CPU リソース制限設定の削除手順

#### 4-1.CPU リソース制限パラメーターの削除

## 4-1-1.プロジェクトの場合の設定値の削除

CPUリソース制限設定用にプロジェクトに設定したパラメーターの削除手順について説明します。

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) 現在の設定を確認します。

(例として、user.root の CPU キャップ設定とリソースプールの指定を削除します。)

3) CPU キャップ設定を削除します。

## # projmod -r -K project.cpu-cap user.root

■ 実行時に指定しているリソースプール(pool\_1)が存在しない場合や停止している場合は、下記のエラーが表示されコマンドは失敗します。その場合は、リソースプール構成が起動していることを確認してください。

「projmod: Validation error on line 2, project.pool: pools not enabled or pool does not exist: "pool\_1"」

4) リソースプールの指定を削除します。

#### # projmod -r -K project.pool user.root

5)設定を確認します。

```
# projects -l user.root
user.root
projid : 1
comment: ""
users : (none)
groups : (none)
attribs:
```

「attribs:」に何も表示されないことを確認します。



#### 4-1-2.ゾーンの場合の設定値の削除

CPU リソース制限設定用にゾーンに設定したパラメーターの削除手順について説明します。

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) 現在の設定を確認します。

(例として、リソースプールと CPU キャップ設定を削除します。)

```
# zonecfg -z zone01
zonecfg:zone01> export
create -b
set zonepath=/export/zones/zone01
set brand=solaris
set autoboot=false
set pool=pool_1
                                   ← ★リソースプールの指定
set ip-type=exclusive
add anet
set linkname=net0
set lower-link=auto
set configure-allowed-address=false
set link-protection=mac-nospoof
set mac-address=random
set auto-mac-address=2:8:20:f8:0:de
end
add rctl
set name=zone.cpu-cap
                                    ← ★CPU キャップの指定
add value (priv=privileged,limit=800,action=deny)
end
```

3) リソースプールの指定を削除します。

## zonecfg:zone01> set pool= ""

- ☞ ダブルクォーテーション(``)内で何も指定せず実行します。
- 4) CPU キャップのパラメーターを削除します。

#### zonecfg:zone01> remove rctl name=zone.cpu-cap

5) 設定を確認します。

```
zonecfg:zone01> export
create -b
set zonepath=/export/zones/zone01
set brand=solaris
set autoboot=false
set ip-type=exclusive
add anet
set linkname=net0
```



set lower-link=auto set configure-allowed-address=false set link-protection=mac-nospoof set mac-address=random set auto-mac-address=2:8:20:f8:0:de end

- set pool パラメーターと set rctl パラメーターが削除されたことを確認します。
- 6) 構成を保存して終了します。

zonecfg:zone01> commit
zonecfg:zone01> exit

7) ゾーンを起動(または再起動)します。

# zoneadm -z zone01 boot [ or reboot ]

### 4-2.リソースプール構成の削除

- 1) global zone に一般ユーザーでログイン後、root の役割を引き受けます。
- 2) リソースプールの構成を確認します。(例として、pool\_1を削除します。)

3) リソースプール構成定義を変更します。(例として、pool\_1を削除します。)

# poolcfg -c 'destroy pool pool 1'

4) リソースプール構成定義を反映します。

#### # pooladm -c

- pool\_1 をリソースプールとして使用中のプロジェクトやゾーンは、上記コマンドの実行直後に pool\_default に再結合されます。
- プロジェクトやゾーン側でリソースプールを指定している場合は、先にそちらの設定を削除した後、リソースプールを削除してください。
- 5) リソースプールの構成を確認します。

# poolstat -r all
id pool type rid rset min max size used load
0 pool\_default pset -1 pset\_default 1 66K 8 0.00 0.00

pool\_1 が表示されないことを確認します。



# 関連 URL

1-1-1. Oracle Solaris 11 ゾーンを使ってみよう【Technical Park】

https://www.fujitsu.com/jp/sparc-technical/document/solaris/#solaris-zone



# 改版履歴

改版年月	版数	改版内容
2012.11	1.0	新規作成
2019.09	1.1	関連 URL の変更