|  |
| --- |
| トライテック |
| 起動・停止・再起動 |
|  |
|  |
| **Noritaka WATANABE** |
| 第1版 2015/01/02  第2版 2015/08/27  第３版　2016/02/09 |
|  |
|  |

目次

[はじめに 2](#_Toc442782638)

[1. Paxos通番 2](#_Toc442782639)

[2. 起動アルゴリズム 2](#_Toc442782640)

[2.1. Synchronize 2](#_Toc442782641)

[2.2. Majorityと合意 3](#_Toc442782642)

[3. 起動手順 3](#_Toc442782643)

[4. API 5](#_Toc442782644)

[4.1. 構造体 5](#_Toc442782645)

[4.1.1. Shudown要求 5](#_Toc442782646)

[4.1.2. Paxos開始モード 5](#_Toc442782647)

[4.2. 関数 5](#_Toc442782648)

[4.2.1. PaxosStart関数 5](#_Toc442782649)

[4.2.2. PaxosAcceptorStartSync関数 6](#_Toc442782650)

[4.2.3. Shutdownコールバック関数 6](#_Toc442782651)

[5. 使用例 6](#_Toc442782652)

[5.1. 配備と初期化(css\_deploy.sh) 6](#_Toc442782653)

[5.2. 開始(css\_start.sh) 9](#_Toc442782654)

[5.3. 停止(css\_stop.sh) 12](#_Toc442782655)

[5.4. 再開始(css\_restart.sh) 14](#_Toc442782656)

|  |
| --- |
|  |

# はじめに

　Paxosセルの起動・停止、障害、再起動について説明する。

# Paxos通番

　Paxosの各合意インスタンスには昇順の通番が付与され、無限の合意列となる。

Multi-Paxosを実現するにあたり、この無限の合意列（決定、未決定を含む）全体についてPaxos合意を行うと考える。

全てのサーバ（レプリカ）は、それぞれ合意列を昇順に非同期に実行する。放っておくと先行するmajorityと遅れるminorityの分割が生じ、たとえ同一性能のサーバであっても、差が拡大する現象が発生する。これを回避するために追いつく（catchup）ための機能が必要となる。

　また、障害から参入（join）する場合には、状態のレプリケートによる同期（synchronize）を行い、それ以降はcatchupあるいは合意に参加する。

1. 合意列

0

∞

②Catchup

A

B

Catchup

Catchup　error

③Synchronize

A

B

Synchronize

# 起動アルゴリズム

## Synchronize

　（状態、アクション列）によるレプリケーションは最新通番に至る先行通番を順次取得すればよい。

　先行者は次通番（次に決定すべき通番）有しているものとし、以下のようにする。

1. 立ち上げ時に次通番を持っていないか、持っているが先行者の方が新しい場合には先行者にシンクロナイズする。
2. 持っていて先行者より新しければそのまま立ち上がる。先行者は立ち上げサーバにcatchupするが立ち上げサーバには最新通番までのデータがないのでエラー終了する。このサーバは次通番を持たないサーバとして再立ち上げをする。
3. また、次通番を持っていないサーバは先行者がいない限り立ち上がらない。したがって、立ち上がったサーバは全て次通番を有していることになる。つまり、次通番-1まで合意済みである。

## Majorityと合意

　前節は合意済みの通番の同期であった。Paxosではmajority（過半数）による合意で通番が進められる。再参入についての合意形成について検討する。

まず、前提として

1. 合意はmajorityが行う。（無限の合意列についての合意である）
2. Majorityは最新通番を有するとともにそれまでの通番は合意済みとなる。

∵）（新）majorityは他の（旧）majorityを継承する。つまり、majority間には少なくとも一つの共通サーバがあり、他のサーバはこれに同期している。

とする。

1. 再起動サーバsがあるグループGに参加しそのグループが新しくmajority　Bとなった場合は、グループのサーバあるいは再起動サーバが異なる（旧）majority　Aのメンバである。
2. グループのサーバが異なるmajority　Aのメンバであれば最新通番を持っているのでsは最新通番に追随できる。
3. グループのサーバ全てがmajority　Aのメンバでなければ再起動サーバのみがAのメンバとなる。
   1. Aからsを除いたときにはAはminorityとなり合意が停止している。
   2. したがって、sは最新通番を有する。

　これから、再起動サーバが通番を有する場合には最新通番に追随でき、合意に参加してよいことになる。

　実は、これはシステム稼働中の通信障害によるネットワーク分割からの再接続による復旧と同じ現象である。

　障害等からの復旧の場合には通番を持っていないのでGの最新通番に追随するが、合意には参加しないようにする。s以外で決定された通番がsに通知された時点で合意に参加するようにする。つまり、s以外ですでにmajorityが構成されていることになるので、合意に再参加できるのである。これを待たないとsは不定のままmajorityを構成する場合が発生する。

# 起動手順

起動手順は次の通りである。

　ここで、各サーバの固有情報として通番が記録されていなければならない。

また、最新の通番を持つサーバを最初に起動する。これは、不必要なcatchupエラーを避けるためである。

なお、どのような順番で起動されたとしても、majorityが構成された時には最新通番が含まれている。（前節から）

最新通番のｻｰﾊﾞから起動

先行次通番＞次通番

シンクロナイズ

次通番なし

合意抑止

合意解除

Start

合意

Yes

No

Yes

No

# API

## 構造体

### Shudown要求

　「PaxosSessionShutdown cell id comment」で強制正常終了させる時の構造体でcallback関数shutdownの引数に渡される。

typedef struct PaxosSessionShutdownReq {

PaxosSessionHead\_t s\_Head;

char s\_Comment[COMMENT\_SIZE];

} PaxosSessionShutdownReq\_t;

### Paxos開始モード

　障害からの立ち上げ（PAXOS\_WAIT）、参入起動（PAXOS\_NO\_SEQ）、初期起動/再起動（PAXOS\_SEQ）がある。障害からの立ち上げの場合には次通番の合意形成には参加しない。再起動時の通番は、「PaxosSessionShutdown」コマンドでサーバーで記録される。

|  |  |  |
| --- | --- | --- |
| 初期起動 | 通番0でPAXOS\_SEQ |  |
| 再起動 | Shutdown時の最新通番でPAXOS\_SEQ | 自分より新しい通番で他が起動された場合にはキャッチアップエラーとなる。この時は、参入起動でやり直す。 |
| 参入起動 | PAXOS\_NO\_SEQ | 先行者がいなければエラー |
| 障害起動 | PAXOS\_WAIT | 先行者がいなければエラー  最初の合意形成に参入しない |

typedef enum PaxosStartMode {

PAXOS\_WAIT = 0, // synchronize and wait decision

PAXOS\_NO\_SEQ, // synchronize and join into decision

PAXOS\_SEQ // synchronize/restart and join into decision

} PaxosStartMode\_t;

## 関数

### PaxosStart関数

　Paxosレベルの開始関数である。

実際には、次のSessionモジュールの関数を使用する。この関数がPaxos開始関数を呼び出す。

extern int PaxosStart( struct Paxos\* pPaxos,

PaxosStartMode\_t StartMode, uint32\_t nextDecide );

pPaxos Paxos構造体

StartMode 開始モード

nextDecide PAXOX\_SEQの時の次通番

### PaxosAcceptorStartSync関数

　Sessionレベルの開始関数である。

本関数は、シンクロナイズすべきサーバを探し出し、シンクロナイズ後PaxosStartを呼び出す。シンクロナイズ直前には、fInitSyncのコールバックが呼び出されるのでAPで必要な初期化を行う。

　PAXOS\_WAIT、PAXOS\_NO\_SEQの時、先行者が見つからなければエラーとなる。

extern int PaxosAcceptorStartSync( struct PaxosAcceptor\* pAcceptor,

PaxosStartMode\_t StartMode, uint32\_t nextDecide,

int (\*fInitSync)(int tartget) );

pAcceptor Acceptor構造体

StartMode 開始モード

nextDecide 次通番

### Shutdownコールバック関数

　本関数は、「PaxosSessionShutdown」でshudown通知を受けた時に呼び出される。APは次通番を個別に記録する。

etern int (\*if\_fShutdown)(struct PaxosAcceptor\*pAcceptor,

PaxosSessionShutdownReq\_t\*pReq, uint32\_t nextDecide);

pAcceptor Acceptor構造体

pReq shutdown要求構造体

nextDecide 次通番

# 使用例

　セルに対する操作についてCssを例に説明する。

## 配備と初期化(css\_deploy.sh)

　構成定義ファイルを参照し、ロードモジュールの配備と環境の初期化を行う。

#!/bin/sh

F\_LOGDIR="~/PAXOS/NWGadget/bin"

F\_PAXOSDIR="~/PAXOS/Paxos\_2/paxos/bin"

F\_SESSIONDIR="~/PAXOS/Paxos\_2/session/bin"

F\_PFSDIR="~/PAXOS/Paxos\_2/PFS/bin"

eval F\_LOGDIR=$F\_LOGDIR

eval F\_SDIR=$F\_SDIR

eval F\_PAXOSDIR=$F\_PAXOSDIR

eval F\_SESSIONDIR=$F\_SESSIONDIR

eval F\_PFSDIR=$F\_PFSDIR

CMDS\_LOG="$F\_LOGDIR/LogFiles $F\_LOGDIR/LogPrint"

CMDS\_PAXOS="$F\_PAXOSDIR/PaxosAdmin"

CMDS\_SESSION="$F\_SESSIONDIR/PaxosSession\*"

CMDS\_PFS="$F\_PFSDIR/PFS\*"

if [ $# -lt 1 ]; then

echo "USAGE:css\_deploy service"

exit 1

fi

service=$1

conf=$HOME/\_$service.conf

while read id paxos udp host session bin root

do

echo "[$id $host $session $paxos $udp $bin $root]"

if [ ! $id ]; then break; fi

if [ $id = "#" ]; then continue; fi

#

# Deploy bin

#

T\_PFS=$bin

echo "ssh -t $host mkdir -p $T\_PFS"

sh <<EOT

ssh -t -t $host mkdir -p $T\_PFS

EOT

echo "scp $CMDS\_PFS $host:$T\_PFS"

sh <<EOT

scp $CMDS\_PFS $host:$T\_PFS

EOT

echo "scp $CMDS\_PAXOS $host:$T\_PFS"

sh <<EOT

scp $CMDS\_PAXOS $host:$T\_PFS

EOT

echo "scp $CMDS\_SESSION $host:$T\_PFS"

sh <<EOT

scp $CMDS\_SESSION $host:$T\_PFS

EOT

echo "scp $CMDS\_LOG $host:$T\_PFS"

sh <<EOT

scp $CMDS\_LOG $host:$T\_PFS

EOT

#

# Deploy conf

#

echo "scp $conf $host:~"

sh <<EOT

scp $conf $host:~

EOT

echo "ssh -t $host rm -rf $root"

sh <<EOT

ssh -t -t $host rm -rf $root

EOT

echo "ssh -t $host mkdir -p $root/PFS"

sh <<EOT

ssh -t -t $host mkdir -p $root/PFS

EOT

echo "ssh -t $host mkdir -p $root/DATA"

sh <<EOT

ssh -t -t $host mkdir -p $root/DATA

EOT

done < $conf

exit 0

## 開始(css\_start.sh)

　構成定義ファイルを参照し、未起動のサーバーを起動する。

#!/bin/sh

F\_LOGDIR="~/PAXOS/NWGadget/bin"

F\_PAXOSDIR="~/PAXOS/Paxos\_2/paxos/bin"

F\_SESSIONDIR="~/PAXOS/Paxos\_2/session/bin"

F\_PFSDIR="~/PAXOS/Paxos\_2/PFS/bin"

eval F\_LOGDIR=$F\_LOGDIR

eval F\_SDIR=$F\_SDIR

eval F\_PAXOSDIR=$F\_PAXOSDIR

eval F\_SESSIONDIR=$F\_SESSIONDIR

eval F\_PFSDIR=$F\_PFSDIR

CMDS\_LOG="$F\_LOGDIR/LogFiles $F\_LOGDIR/LogPrint"

CMDS\_PAXOS="$F\_PAXOSDIR/PaxosAdmin"

CMDS\_SESSION="$F\_SESSIONDIR/PaxosSession\*"

CMDS\_PFS="$F\_PFSDIR/PFS\*"

if [ $# -lt 1 ]; then

echo "USAGE:css\_deploy service"

exit 1

fi

service=$1

conf=$HOME/\_$service.conf

while read id paxos udp host session bin root

do

echo "[$id $host $session $paxos $udp $bin $root]"

if [ ! $id ]; then break; fi

if [ $id = "#" ]; then continue; fi

#

# Deploy bin

#

T\_PFS=$bin

echo "ssh -t $host mkdir -p $T\_PFS"

sh <<EOT

ssh -t -t $host mkdir -p $T\_PFS

EOT

echo "scp $CMDS\_PFS $host:$T\_PFS"

sh <<EOT

scp $CMDS\_PFS $host:$T\_PFS

EOT

echo "scp $CMDS\_PAXOS $host:$T\_PFS"

sh <<EOT

scp $CMDS\_PAXOS $host:$T\_PFS

EOT

echo "scp $CMDS\_SESSION $host:$T\_PFS"

sh <<EOT

scp $CMDS\_SESSION $host:$T\_PFS

EOT

echo "scp $CMDS\_LOG $host:$T\_PFS"

sh <<EOT

scp $CMDS\_LOG $host:$T\_PFS

EOT

#

# Deploy conf

#

echo "scp $conf $host:~"

sh <<EOT

scp $conf $host:~

EOT

echo "ssh -t $host rm -rf $root"

sh <<EOT

ssh -t -t $host rm -rf $root

EOT

echo "ssh -t $host mkdir -p $root/PFS"

sh <<EOT

ssh -t -t $host mkdir -p $root/PFS

EOT

echo "ssh -t $host mkdir -p $root/DATA"

sh <<EOT

ssh -t -t $host mkdir -p $root/DATA

EOT

done < $conf

exit 0

nw@paxos00:~/PAXOS/Paxos\_2/PFS/bin$ cat css\_start.sh

#!/bin/sh

if [ $# -lt 1 ]

then

echo "USAGE:css\_start.sh service"

exit 1

fi

env="LOG\_SIZE=20000 "

files=20

service=$1

conf=$HOME/\_$service.conf

#

# Up PFSServer

#

while read id paxos udp host session bin root

do

if [ ! $id ]; then break; fi

if [ $id = "#" ]; then continue; fi

echo "ssh -f $host \"cd $root;$env $bin/PFSServer $service $id $root TRUE 0 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files\""

ssh -f $host "cd $root;$env $bin/PFSServer $service $id $root TRUE 0 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files"

sleep 1

done < $conf

exit 0

## 停止(css\_stop.sh)

　構成定義ファイルを参照し、稼動中のサーバーを停止する。ただし、リーダーがいる場合には、その他をABORTさせ、最後にリーダーを正常停止する。正常停止の場合、「SHUTDOWN」ファイルに次通番とコメントが書き込まれる。

#!/bin/sh

if [ $# -lt 2 ]

then

echo "USAGE:css\_stop.sh service comment"

exit 1

fi

service=$1

conf=$HOME/\_$service.conf

comment=$2

while read id paxos udp host session bin root

do

if [ ! $id ]; then break; fi

if [ `echo $id|cut -c1` = '#' ]; then continue; fi

ssh -n $host "$bin/PaxosAdmin $service $id is\_active > /dev/null 2>&1"

ret=$?

if [ $ret = 0 ]; then

ssh -n $host "$bin/PaxosAdmin $service $id is\_leader > /dev/null 2>&1"

ret=$?

if [ $ret = 0 -a ! $leader\_id ]; then

leader\_id=$id

leader\_host=$host

leader\_bin=$bin

else

IDs=$IDs"$id "

eval bin\_$id=\$bin

eval root\_$id=\$root

eval host\_$id=\$host

fi

fi

done < $conf

for id in $IDs

do

eval host='$host\_'$id

eval bin='$bin\_'$id

eval root='$root\_'$id

shutdown=$bin/PaxosSessionShutdown

if [ $leader\_id ]; then

echo "ssh -f $host $shutdown $service $id ABORT"

ssh -f $host $shutdown $service $id ABORT

else

echo "ssh -f $host $shutdown $service $id $comment"

ssh -f $host $shutdown $service $id $comment

fi

done

if [ $leader\_id ]; then

shutdown=$leader\_bin/PaxosSessionShutdown

echo "ssh -f $leader\_host $shutdown $service $leader\_id $comment"

ssh -f $leader\_host $shutdown $service $leader\_id $comment

fi

exit 0

## 再開始(css\_restart.sh)

　構成定義ファイルを参照し、未稼働のサーバーを起動する。ただし、最新通番があればこれを最初に起動する。

#!/bin/sh

if [ $# -lt 1 ]

then

echo "USAGE:css\_restart.sh service"

exit 1

fi

env="LOG\_SIZE=20000 "

files=20

service=$1

conf=$HOME/\_$service.conf

#

# Check active

#

max=$((-1))

while read id paxos udp host session bin root

do

if [ ! $id ]; then break; fi

if [ `echo $id|cut -c1` = '#' ]; then continue; fi

ssh -n $host "$bin/PaxosAdmin $service $id is\_active > /dev/null 2>&1"

ret=$?

if [ $ret != 0 ]

then

IDs=$IDs"$id "

eval bin\_$id=\$bin

eval root\_$id=\$root

eval host\_$id=\$host

scp $host:$root/SHUTDOWN /tmp/SHUTDOWN > /dev/null 2>&1

ret=$?

if [ $ret = 0 ]

then

read seq comment < /tmp/SHUTDOWN

num=$((seq))

if [ $max -eq -1 ] # first

then

max=$num; young=$id

elif [ $max -lt $num ]

then

diff=$((num-max))

if [ $diff -lt 2147483647 ]

then

max=$num; young=$id

fi;

else

diff=$((max-num))

if [ $diff -ge 2147483647 ]

then

max=$num; young=$id

fi;

fi

eval seq\_$id=$seq

eval comment\_$id=$comment

fi

fi

done < $conf

#

# Start youngest

#

if [ $young ]

then

eval bin='$bin\_'$young

eval root='$root\_'$young

eval host='$host\_'$young

echo "ssh -f $host \"cd $root;$env $bin/PFSServer $service $young $root TRUE $max 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files\""

ssh -f $host "cd $root;$env $bin/PFSServer $service $young $root TRUE $max 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files"

sleep 1

fi

for id in $IDs

do

if [ $young ]; then

if [ $young = $id ]; then continue; fi

fi

eval host='$host\_'$id

eval bin='$bin\_'$id

eval root='$root\_'$id

eval seq='$seq\_'$id

if [ $seq ]

then

echo "ssh -f $host \"cd $root;$env $bin/PFSServer $service $id $root TRUE $seq 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files\""

ssh -f $host "cd $root;$env $bin/PFSServer $service $id $root TRUE $seq 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files"

else

if [ $young ]

then

echo "ssh -f $host \"cd $root;$env $bin/PFSServer $service $id $root TRUE 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files\""

ssh -f $host "cd $root;$env $bin/PFSServer $service $id $root TRUE 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files"

else

echo "ssh -f $host \"cd $root;$env $bin/PFSServer $service $id $root FALSE 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files\""

ssh -f $host "cd $root;$env $bin/PFSServer $service $id $root FALSE 3>&1 1>&2 2>&3 | $bin/LogFiles CSS $files"

fi

fi

sleep 1

done

exit 0