kumofs

FURUHASHI Sadayuki

Contents

Part I

kumofs user guide

Chapter 1

kumofsとは

kumofs は key-value 型のデータを保存する非常に高速な分散ストレージです。1 台か 2 台の Manager ノードと、複数の Server ノードでクラスタを構成します。動作中にサーバーを追加することができ、読み・書き両方の性能をスケールアウトさせることができます。またデータのレプリケーションをサポートしており、一部のサーバーに障害が発生しても正常に動作し続けます。memcached 互換のプロトコルをサポートしており、テキスト・バイナリどちらのプロトコルも利用できます。

Chapter 2

インストール

kumofs をコンパイルして実行するには以下の環境が必要です：

- Linux >= 2.6.18
- Tokyo Cabinet >= 1.4.10
- MessagePack for C++ >= 0.3.1
- MessagePack for Ruby >= 0.3.1
- ruby >= 1.8.7
- libcrypto (openssl)
- zlib
- g++ >= 4.1（コンパイルのみ）
- ragel >= 6.3 (コンパイルのみ)

./configure && make && make install でインストールできます。

```
$ ./configure
$ make
$ sudo make install
```

2.1 configure フラグ

–**with-tokyocabinet=DIR** Tokyo Cabinet がインストールされているディレクトリを指定する

–**with-msgpack=DIR** MessagePack がインストールされているディレクトリを指定する

–**with-tcmalloc**[**=DIR**] tcmalloc とリンクする

–**enable-trace** 画面を埋め尽くすほど冗長なデバッグ用のメッセージを出力するようにする

Chapter 3

チュートリアル

kumofs は、実際にデータを保存する **Server** ノード、Server ノード群を管理する **Manager** ノード、アプリケーションからのリクエストを中継する **Gateway** の 3 種類のプロセスから構成されます。それぞれ **kumo-server**、**kumo-manager**、**kumo-gateway** というコマンドが対応します。

kumo-gateway は memcached プロトコルをサポートしています。kumofs を利用したいホストで kumo-gateway を動作させておくことで、アプリケーションは localhost の kumo-gateway に memcached クライアントライブラリを使って接続することで kumofs を利用できます。

たとえば **s1**, **s2**, **s3**, **s4** の 4 台のホストを Server ノード、**m1** を Manager ノードとし、**c1**, **c2** の 2 台から利用するには、以下のようにします：

```
[m1]$ kumo-manager -v -l m1
[s1]$ kumo-server  -v -m m1 -l s1 -s database.tch    # -m で Manager を指定する
[s2]$ kumo-server  -v -m m1 -l s2 -s database.tch    # -l は自ホストのアドレス
[s3]$ kumo-server  -v -m m1 -l s3 -s database.tch    # -s はデータベース名
[s4]$ kumo-server  -v -m m1 -l s4 -s database.tch    # -v は冗長なメッセージを出力
[c1]$ kumo-gateway -v -m m1 -t 11211    # 11211/tcp で memcached テキストプロトコル
[c2]$ kumo-gateway -v -m m1 -t 11211    # を待ち受ける
```

kumo-manager と kumo-server と kumo-gateway を同じホスト上で動かすこともできます。たとえば **m1** と **s1** と **c1** は実際には同じホストでも大丈夫です。

kumo-manager は 2 台のホストで冗長化することができます。**m1** と **m2** の 2 台のホストで kumo-manager を起動するには以下のようにします：

```
[m1]$ kumo-manager -v -l m1 -p m2    # Manager 同士は互いに指定する
[m2]$ kumo-manager -v -l m2 -p m1    # Manager 同士は互いに指定する
[s1]$ kumo-server  -v -m m1 -p m2 -l s1 -s database.tch    # -m と-p で Manager を指定する
[s2]$ kumo-server  -v -m m1 -p m2 -l s2 -s database.tch
[s3]$ kumo-server  -v -m m1 -p m2 -l s3 -s database.tch
[s4]$ kumo-server  -v -m m1 -p m2 -l s4 -s database.tch
[c1]$ kumo-gateway -v -m m1 -p m2 -t 11211    # -m と-p で Manager を指定する
[c2]$ kumo-gateway -v -m m1 -p m2 -t 11211    # 11211/tcp で memcached テキストプロトコル
```

すべてのプロセスを **localhost** で動かすには以下のようにします：

```
[localhost]$ kumo-manager -v -l localhost    # Server はポートを変更して起動する
[localhost]$ kumo-server  -v -m localhost -l localhost:19801 -L 19901 -s database1.tch
[localhost]$ kumo-server  -v -m localhost -l localhost:19802 -L 19902 -s database2.tch
[localhost]$ kumo-gateway -v -m localhost -t 11211    # 11211/tcp で memcached テキストプロトコル
```

3.1 Server ノードの追加

動作中に Server ノードを追加するには、新しく kumo-server を起動し、**kumoctl** コマンドを使って登録します。kumoctl コマンドには Manager（2 台動作しているならどちらか片方）のアドレスを指定します。

```
[s5]$ kumo-server -v -m m1 -p m2 -l s5 -s database.tch    # 新しい Server を起動
[xx]$ kumoctl m1 status    # 新しい Server が認識されているか確認
[xx]$ kumoctl m1 attach    # 新しい Server を attach
```

Chapter 4

障害からの復旧方法

kumofs では、1 つの key-value のペアは 3 台の Server ノードにコピーされます。そのため 2 台までなら Server ノードがダウンしてもデータが欠損することなく動作し続けます。

また Manager ノードは 2 台で冗長構成を取ることができます。片方がダウンしても正常に動作し続け、両方ダウンしてもその間に Server に障害が発生しなければ問題なく動作し続けます。

4.1 Server ノードの復旧

Server ノードが 1 台ダウンすると、一部の key-value ペアの複製が 1 つ減った状態のまま動作し続けます。2 台ダウンすると、1 つか 2 つ減った状態のままになります。この状態から複製の数を 3 つに戻すには、Server ノードを復旧させたあと kumoctl コマンドを使って登録するか、Server ノードを復旧させずにダウンした Server ノードを完全に切り離します。

まず kumoctl コマンドを使ってどの Server ノードに障害が発生しているかを確認します：

```
[xx]$ kumoctl m1 status    # Manager のアドレスを指定して状態を取得
hash space timestamp:
  Wed Dec 03 22:15:35 +0900 2008 clock 50
attached node:
  192.168.0.101:19800  (active)
  192.168.0.102:19800  (active)
  192.168.0.103:19800  (active)
  192.168.0.104:19800  (fault)
not attached node:
```

(fault) と表示されている Server ノードに障害が発生しています。ここで Server ノードを再起動すると、以下のようになります：

```
[xx]$ kumoctl m1 status
hash space timestamp:
  Wed Dec 03 22:15:45 +0900 2008 clock 58
attached node:
  192.168.0.101:19800  (active)
  192.168.0.102:19800  (active)
  192.168.0.103:19800  (active)
  192.168.0.104:19800  (fault)
not attached node:
  192.168.0.104:19800
```

not attached node のところに表示されている Server ノードは、Manager ノードから認識されているが、まだ登録されていない Server ノードの一覧です。

ここで **attach** コマンドを発行すると、Server ノードが実際に登録され、データの複製が 3 つになるようにコピーされます：

```
[xx]$ kumoctl m1 attach    # attach
[xx]$ kumoctl m1 status
hash space timestamp:
  Wed Dec 03 22:15:55 +0900 2008 clock 62
attached node:
  192.168.0.101:19800  (active)
  192.168.0.102:19800  (active)
  192.168.0.103:19800  (active)
  192.168.0.104:19800  (active)
not attached node:
```

attach コマンドを発行して新たに Server ノードを登録するとき、データの複製をコピーするための比較的大規模なネットワークトラフィックが発生することに注意してください。

kumoctl コマンドで、attach コマンドの代わりに **detach** コマンドを発行すると、fault 状態の Server ノードが切り離されます。このときも複製の数が 3 つになるようにデータのコピーが行われます。

```
[xx]$ kumoctl m1 detach    # detach
[xx]$ kumoctl m1 status
hash space timestamp:
  Wed Dec 03 22:15:55 +0900 2008 clock 62
attached node:
  192.168.0.101:19800  (active)
  192.168.0.102:19800  (active)
  192.168.0.103:19800  (active)
not attached node:
  192.168.0.104:19800
```

FIXME Server ノードを復旧するときにのデータベースファイルの扱い delete したデータが復活する件

4.2 Manager ノードの復旧

2 台で冗長構成を取っているときに片方の Manager ノードがダウンした場合は、Manager ノードを再起動してください。Manager ノードの IP アドレスは障害発生前と同じにしておく必要があります。

両方の Manager ノードがダウンした場合や、Manager ノードを 1 台で運用していた場合は、Manager ノードを再起動し、kumoctl コマンドで attach コマンドを発行してください。

Chapter 5

チューニング

5.1 Tokyo Cabinet のチューニング

Tokyo Cabinet のハッシュデータベースのチューニングによって性能は大きく変わります。kumo-server を起動する前に、tchmgr コマンドを使ってデータベースファイルをあらかじめ作成しておいてください。tchmgr コマンドのパラメータについては、Tokyo Cabinet のドキュメントを参照してください。Tokyo Cabinet のパラメータのうち、拡張メモリマップのサイズ（xmsiz）とキャッシュ機構（rcnum）は kumo-server のコマンドライン引数で指定します。kumo-server の**-s** オプションで、データベースファイル名の後ろに**#xmsiz=XXX** と指定すると拡張メモリマップのサイズを指定できます。**#rcnum=XXX** と指定するとキャッシュ機構を有効化できます。

```
[s2]$ kumo-server  -v -m m1 -l s2 -s database.tch#xmsiz=600m#rcnum=4k
```

5.2 スレッド数

CPU のハードウェアスレッドの数が多い場合は、kumo-server と kumo-gateway のワーカースレッドの数（-TR 引数）を増やすと性能が向上します。ハードウェアスレッド数+2 くらいが目安です。デフォルトは 4 です。保存する 1 つ 1 つの key-value ペアのサイズが大きい場合は、kumo-server と kumo-gateway の送信用スレッドの数（-TW 引数）を増やすと性能が向上する可能性があります。デフォルトは 2 です。

5.3 非同期レプリケーション

kumofs ではデータを set したり delete したりするときレプリケーションを行いますが、デフォルトではレプリケーションが完了するまで待ってから（すべてのサーバーから応答が帰ってきてから）アプリケーションに応答が返されます。これを 1 台の Server に書き込みが完了した時点で応答を返すようにすると（非同期レプリケーション）、更新系の応答時間が大幅に短縮されます。ただし非同期レプリケーションを有効にすると、成功応答が帰ってきたとしても、必ずしもレプリケーションが成功していることが保証されず、そのため複数の Server ノード間でデータの一貫性が保たれているとが保証されなくなります。Set 操作のレプリケーションを非同期にするには、kumo-gateway のコマンドライン引数に**-As** を、Delete 操作のレプリケーションを非同期にするには**-Ad** を追加してください。

Chapter 6

リファレンス

6.1 保証範囲の制限

kumofs はスケーラビリティとアベイラビリティを持つ代わりに、複数の Server ノード間に保存されるデータの一貫性に制限があります。アプリケーションで以下に挙げる保証範囲に含まれない動作を前提としないように注意してください。

6.1.1 Set

Set 操作は key-value ペアを保存します。Set が成功応答を返した場合は一貫性は保証されます。つまりすべての担当 Server ノードに同じ key-value ペアが書き込まれ、どのノードからでも同じ値を直後に Get することができ、古い値が読み出されることはありません。Set が失敗応答を返した場合は一貫性は保証されません。つまり Server ノードによって古い値を持っていたり新しい値を持っていたりします。Set 操作を実行中に同じ key を Get した場合は、新旧どちらの値が読み出されるかは不定ですが、どちらかの値が読み出されます。混ざった値が読み出されることはありません。

6.1.2 Delete

Delete 操作は key-value ペアを削除します。Delete 操作は一貫性を保証しませんが、できる限り一貫性が保たれるように努力します。Delete 操作は実際には「delete フラグ」を Set する操作です。しかし delete フラグは時間が経過すると回収され、本当に削除されます。delete フラグが回収されると一貫性は保証されません。delete フラグは以下の条件で回収されます：

Delete してから一定の時間が経過した この時間は kumo-server の**-gX** オプションで指定できます。

Delete フラグを記憶するメモリ使用量の上限に達し、かつ Delete してから一定の時間が経過した この時間は kumo-server の**-gN** オプションで指定できます。メモリ使用量の上限は**-gS** オプションで指定できます。

delete フラグを記憶するメモリ使用量の上限に達したが、Delete してから一定の時間が経過していない場合は、delete フラグはデータベースファイルの中に放置されます。放置された delete フラグは、次に再配置操作が行われたときに回収されます。

6.2 ログ

kumo-manager, kumo-server, kumo-gateway は、それぞれ 2 種類のログを出力します：

テキストログ 行区切りのテキストフォーマットのログ。標準出力に出力される

バイナリログ MessagePack でシリアライズされたバイナリ形式のログ

テキストログは常に出力されます。**-v** オプションを付けると冗長なログも出力されるようになります。テキストログはファイルに書き出すこともできるが、ログローテーションはサポートしていません。デフォルトでは優先度によってログに色が付きますが、**-d** <**path.pid**> オプションを指定してデーモンとして起動するか、**-o** "**-**"オプションを指定すると、ログに色が付かなくなります。

バイナリログは**-g** <**path.mpac**> オプションを付けたときだけ出力されます。バイナリログは SIGHUP シグナルを受け取るとログファイルを開き直すため、logrotate などを使ってログローテーションができます。

6.3 共通のコマンドライン引数

-o <**path.log**> ログを標準出力ではなく指定されたファイルに出力する。**-**を指定すると標準出力に出力する。省略するとログに色を付けて標準出力に出力する

-v WARN よりレベルの低いメッセージを出力する

-g <**path.mpac**> バイナリログを指定されたファイルに出力する

-d <**path.pid**> デーモンになる。指定されたファイルに pid を書き出す

-Ci <**sec**> タイマークロックの間隔を指定する。単位は秒で、小数を指定できる

-Ys <**sec**> connect(2) のタイムアウト時間を指定する。単位は秒で、小数を指定できる

-Yn <**num**> connect(2) のリトライ回数を指定する

-TR <**num**> ワーカースレッドの数を指定する

-TW <**num**> 送信用スレッドの数を指定する

6.4 kumo-manager

-l <**address**> 待ち受けるアドレス。**他のノードから見て**接続できるホスト名とポート番号を指定する

-p <**address**> もし存在するなら、もう一台の kumo-manager のホスト名とポート番号を指定する

-c <**port**> kumoctl からのコマンドを受け付けるポート番号を指定する

-a Server が追加・離脱されたときに、マニュアル操作を待たずにレプリケーションの再配置を自動的に行うようにする。実行中でも kumoctl コマンドを使って変更できる

-Rs 自動的な再配置が有効なときに、サーバーの追加・離脱を検出してからレプリケーションの再配置を開始するまでの待ち時間を指定する。単位は秒

6.5 kumo-server

-l <**address**> 待ち受けるアドレス。**他のノードから見て**接続できるホスト名とポート番号を指定する

-L <**port**> kumo-server が待ち受けるもう一つのポートのポート番号を指定する

-m <**address**> kumo-manager のホスト名とポート番号を指定する

-p <**address**> もし存在するなら、もう一台の kumo-manager のホスト名とポート番号を指定する

-s <**path.tch**[**#xmsiz=SIZE**][**#rcnum=SIZE**]> データを保存するデータベースファイルのパスを指定する

-f <dir> レプリケーションの再配置に使う一時ファイルを保存するディレクトリを指定する。データベースファイルのサイズに応じて十分な空き容量が必要

-gS <seconds> delete したエントリのクロックを保持しておくメモリ使用量の上限を KB 単位で指定する

-gN <seconds> delete したエントリのクロックを保持しておく最小時間を指定する。メモリ使用量が上限に達していると、最大時間に満たなくても最小時間を過ぎていれば削除される。

-gX <seconds> delete したエントリのクロックを保持しておく最大時間を指定する

6.6 kumo-gateway

-m <address> kumo-manager のホスト名とポート番号を指定する

-p <address> もし存在するなら、もう一台の kumo-manager のホスト名とポート番号を指定する

-t <port> memcached テキストプロトコルを待ち受けるポート番号を指定する

-b <port> memcached バイナリプロトコルを待ち受けるポート番号を指定する（EXPERIMENTAL）

-G <number> Get 操作の最大リトライ回数を指定する

-S <number> Set 操作の最大リトライ回数を指定する

-D <number> Delete 操作の最大リトライ回数を指定する

-As Set 操作でレプリケーションするとき、レプリケーション完了の応答を待たずに成功を返すようにする

-Ad Delete 操作でレプリケーションするとき、レプリケーション完了の応答を待たずに成功を返すようにする

6.7 kumoctl

kumoctl コマンドは Manager ノードに様々なコマンドを発行するための管理コマンドです。第一引数に Manager ノードのアドレスを指定し、第二引数にサブコマンドを指定します。

```
Usage: kumoctl address[:port=19799] command [options]
command:
   status                     get status
   attach                     attach all new servers and start replace
   attach-noreplace           attach all new servers
   detach                     detach all fault servers and start replace
   detach-noreplace           detach all fault servers
   replace                    start replace without attach/detach
   full-replace               start full-replace (repair consistency)
   backup  [suffix=????????]  create backup with specified suffix
   enable-auto-replace        enable auto replace
   disable-auto-replace       disable auto replace
```

6.7.1 ハッシュ空間の取得

status サブコマンドは、どの Server にデータを保存するかを決定するハッシュ空間を取得します。以下のように表示されます：

```
hash space timestamp:
  Wed Dec 03 22:15:45 +0900 2008 clock 58
attached node:
  192.168.0.101:19800  (active)
  192.168.0.102:19800  (active)
  192.168.0.103:19800  (active)
  192.168.0.104:19800  (fault)
not attached node:
  192.168.0.104:19800
```

hash space timestamp はハッシュ空間を更新した時刻を示しています。**attached node** は登録されている（ルーティング対象の）Server ノードの一覧を示しています。**(active)** はそのノードが利用可能なことを、**(fault)** はそのノードがダウンしていることを示します。**not attached node** は認識しているが登録されていない Server ノードの一覧を示しています。

6.7.2 Server ノードの追加と切り離し

attach サブコマンドは、認識しているが登録されていない Server ノードを実際に登録します。**detach** サブコマンドは、fault 状態の Server ノードを切り離します。**attach-noreplace** サブコマンドは **attach** と同じですが、Server ノードを登録した後に key-value ペアの複製の再配置を行いません。**detach-noreplace** サブコマンドは **detach** と同じですが、Server ノードを切り離した後に複製の再配置を行いません。**replace** サブコマンドは複製の再配置だけを行います。attach-noreplace サブコマンドと detach-noreplace サブコマンドは attach と detach を同時に行いときのみ使用し、すぐに replace サブコマンドを使って再配置を行ってください。再配置を行わないまま長い間放置してはいけません。

6.7.3 バックアップの作成

backup サブコマンドは、データベースファイルのバックアップを作成します。バックアップは認識しているすべての Server ノード上で作成されます。バックアップファイルのファイル名は、元のデータベースファイル名に第三引数で指定した suffix を付けたファイル名になります。suffix を省略するとその日の日付 (YYMMDD) が使われます。作成されたバックアップファイルは、**kumomergedb** コマンドを使って 1 つのファイルにまとめることができます。

6.8 kumomergedb

kumomergedb コマンドを使うと複数のデータベースファイルを 1 つにまとめることができます。第一引数に出力先のファイル名を指定し、第二引数以降にまとめたいデータベースファイルを指定します。

```
$ kumomergedb backup.tch-20090101 \
              server1.tch-20090101 server2.tch-20090101 server3.tch-20090101
```

6.9 kumolog

kumolog コマンドはバイナリ形式のログを人間にとって読みやすいテキストに変換して表示します。

```
kumolog [options] <logfile.mpac>
```

-f, –follow tail -f と同じ効果

-t, –tail 最後の N 個のログだけ表示する（デフォルト: N=10）

-h, –head 最初の N 個のログだけ表示する（デフォルト: N=10）

-n, –lines=[-]N N を指定する

6.10 kumostat

kumostat コマンドを使うと Server ノードの状態を取得することができます。第一引数に Server ノードのホスト名とポート番号を指定し、第二引数にコマンドを指定します：

```
Usage: kumostat server-address[:port=19800] command
       kumostat -m manager-address[:port=19700] command
command:
   pid                          get pid of server process
   uptime                       get uptime
   time                         get UNIX time
   version                      get version
   cmd_get                      get number of get requests
   cmd_set                      get number of set requests
   cmd_delete                   get number of delete requests
   items                        get number of stored items
   rhs                          get rhs (routing table for Get)"
   whs                          get whs (routing table for Set/Delete)"
   hscheck                      check if rhs == whs
   set_delay                    maximize throughput at the expense of latency"
   unset_delay                  minimize latency at the expense of throughput"
```

-m に続いて Manager ノードのアドレスを指定すると、Manager ノードから Server 一覧を取得し、attach されていて active なすべての Server ノードの状態を表示します。

pid kumo-server プロセスの pid

uptime kumo-server プロセスの起動時間（単位は秒）

time kumo-server プロセスが走っているホストの UNIX タイム

version kumo-server のバージョン

cmd_get Gateway ノードからの Get 操作を処理した回数

cmd_set Gateway ノードからの Set 操作を処理した回数

cmd_delete Gateway ノードからの Delete 操作を処理した回数

items データベースに入っているエントリの数

rhs Get に使われるルーティング表

whs Set/Delete に使われるルーティング表

hscheck rhs と whs が同じかどうかチェックします

set_delay 遅延を犠牲にして最大スループットを最大化する

unset_delay 最大スループットを犠牲にして遅延を最小化する

rhs と whs が食い違っている場合は、再配置を実行中か、前回の再配置が失敗している可能性があります。

6.11 kumotop

kumotop コマンドを使うと Server ノードの状態を定期的に更新しながら表示することができます。引数に監視したい Server ノードのアドレスを指定します。Server ノードのアドレスは複数指定できます：

```
Usage: kumotop server-address[:port=19800] ...
       kumotop -m manager-address[:port=19700]
```

-m に続いて Manager ノードのアドレスを指定すると、Manager ノードから Server 一覧を取得し、attach されているすべての Server ノードの状態を表示します。

6.12 kumohash

kumohash コマンドを使うと、ある key がどの Server に保存されるかを計算することができます。

```
Usage: kumohash server-address[:port=19800] ... -- command [options]
       kumohash -m manager-address[:port=19700] command [options]
command:
   hash  keys...              calculate hash of keys
   assign  keys...            calculate assign node
   dump                       dump hash space
```

hash key のハッシュ値を計算する

assign key がどの Server に保存されるかを計算する

dump Consistent Hashing のルーティング表を表示する

Part II

kumofs internals

Chapter 7

分散アルゴリズム

kumofs はどの Server ノードにデータを保存するかを決定するために、Consistent Hashing を利用しています。ハッシュ関数は SHA-1 で、下位の 64 ビットのみを使います。1 台の物理ノードは 128 台の仮想ノードを持ちます。

データを取得するときは、Gateway が key にハッシュ関数を掛けてハッシュ空間から担当 Server ノードを計算し、担当 Server ノードからデータを取得します。取得に失敗したときは、ハッシュ空間上でその次に当たる物理 Server ノードから取得します。それでも失敗したらその次の次の Server ノードから取得します。それでも失敗したら担当 Server に戻ってリトライします。

ハッシュ関数は Gateway で計算することで Server ノードの負荷は下げています。

データを変更するときは、Gateway が key にハッシュ関数を掛けてハッシュ空間から担当 Server ノードを計算し、担当 Server ノードにデータを送信します。取得する場合とは異なり、次の Server にフォールバックすることはありません。

担当 Server ノードは変更操作を受け取ると、データをハッシュ空間上で次の物理 Server ノードと、次の次の物理 Server ノードにも key-value ペアをコピーします。

Server ノードは Gateway からリクエストを受け取ったとき、本当に自分が担当ノードであるかどうかを自分が持っているハッシュ空間を使って確認します。間違っていた場合はリクエストを拒否します。このように必ず特定の担当 Server ノードだけがデータを変更でき、他の Server ノードが同じタイミングで同じ key-value を変更することがないようになっています。

Server ノードを選ぶとき fault フラグがセットされているノードはスキップします。一部の Server ノードがダウンしている状態でも正常なアクセスを続けられるようになっています。

7.1 Server ノードの追加

Chapter 8

死活監視と障害検出

8.1 障害の検出

メッセージを送ろうとしたところ、接続済みのすべての TCP コネクションでエラーが発生し、再接続を試みても失敗して再接続のリトライ回数が上限に達したら、そのノードはダウンしたと判断します。TCP コネクションが切断されただけではダウンしたとは判断せず、メッセージの送信に失敗しても制限回数以内に再接続することができたら再送されます。Server ノードと Manager ノードは常に keep-alive メッセージをやりとりしており、いつもメッセージを送ろうとしている状態になっています。Server ノードがダウンしたらできるだけ早く検出して fault フラグをセットし、正常なアクセスを継続させます。

8.2 新しいノードの検出

Chapter 9

再配置アルゴリズム