サーバレスクラウド基盤ソフトウェアにおけるリソース管理機構

2009SE014 浅岡 大樹　　2010SE067 伊藤 大典

指導教員：宮澤 元

1　はじめに

コンピュータネットワーク経由で様々なサービスを提供するクラウドコンピューティング (クラウド) というコンピュータの利用方法が普及している．クラウドは，クラウド基盤ソフトウェアを用いて複数のサーバを集約することにより実現されている．クラウドの使用形態は，パブリッククラウドとプライベートクラウドに大別される [1]．パブリッククラウドは大規模でリソースの制約が比較的少ないのに対し，プライベートクラウドは，組織内に設置するサーバだけを用いるので，利用できるリソースに制約がある．

プライベートクラウドのリソース不足を解消するためには，サーバとして利用できるリソースを追加する必要がある．しかし，サーバ用のコンピュータを追加するのはコストがかかるので，我々はクラウドのノードとして利用されていないクライアントに着目した．

我々はプライベートクラウド環境において，組織の持つコンピュータリソースを統一的に管理してクラウドサービスを提供するサーバレスクラウド基盤ソフトウェアを開発中である．これを用いることにより，各コンピュータをサーバやクライアントといった特定の目的だけに用いることなく全体として適切なリソース管理を行うクラウド基盤を実現できる．

本稿では，特にサーバレスクラウド基盤ソフトウェアのリソース提供部分であるノードコントローラの設計と実装について述べる．クライアントコンピュータで動作する非クラウド処理用にリソースを予約することで，クラウド処理が非クラウド処理を妨げることなくクライアントコンピュータ上で動作させることができる．

2　サーバレスクラウド基盤ソフトウェア

本節では，従来のクラウド基盤ソフトウェアとサーバレスクラウド基盤ソフトウェアについて，その概要を示す．

2.1　従来のクラウド基盤ソフトウェア

クラウド基盤ソフトウェアは一般に，クラウドコントローラ，クラスタコントローラ，ノードコントローラ，ストレージコントローラから構成される (図 1)．

クラウドコントローラ (CLC)
: クラウド全体の管理を行う．利用者に対して Web 管理画面を提供し，利用者からの要求をクラスタコントローラへ送信する．

クラスタコントローラ (CC)
: ノード群のネットワークや使用可能リソースを管理する．ノードコントローラへ仮想マシンの動作要求を行うが，その際にどのノードに仮想マシンの作成を行うかスケジューリングを行う．

ノードコントローラ (NC)
: クラスタコントローラからの要求を受け仮想マシンを動作させる．また，動作しているノードの使用可能なリソースの情報をクラスタコントローラへ送信する．仮想マシン作成時は，仮想マシン上で動作させる仮想マシンイメージを取得し，キャッシュしておく．仮想マシンの状態をクラスタコントローラへ定期的に送信する．

ストレージコントローラ (SC)
: ストレージコントローラでは，仮想マシンに対してボリュームの提供や管理を行い，ボリュームのスナップショットの管理も行う．

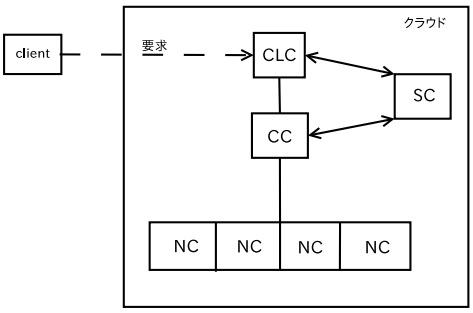

図 1　クラウド基盤ソフトウェアの構成

従来のクラウド基盤ソフトウェアでは，クライアントコンピュータのリソースを使用することは考えられていない．クライアントのリソースをクラウド処理に用いると，クライアントの動作で必要なユーザインタフェース処理などの非クラウド処理のためのリソースが不足するおそれがあるからである．

2.2　サーバレスクラウド基盤ソフトウェア

サーバレスクラウド基盤ソフトウェアは各コンピュータをサーバやクライアントといった特定の目的だけに用いることなく，全てのコンピュータをクラウドのノードとして利用し，ストレージリソースや CPU リソース，メモリリソースについて全体として適切な管理を行うクラウド基盤ソフトウェアである [2]．

サーバレスクラウド基盤ソフトウェアは仮想マシン管理に注目すると，各ノードの利用可能なリソースに応じてノードスケジューリングを行うと共に，各ノードで非クラウド処理に使用するリソースを予約できるようなリソース管理を行う．これにより，非クラウド処理がリソース不足になることを防ぐ．図 2 にサーバレスクラウド環境の構成

を示す．図 1 でクラウドの外にあったクライアントが，クラウド内でノードとして利用されている．

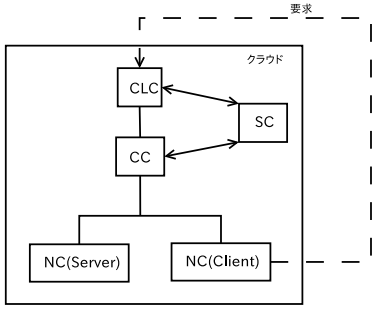

図 2 サーバレスクラウド基盤ソフトウェアの構成

3 リソース管理機構の設計

サーバレスクラウド環境では，全てのノードに自由に仮想マシンを配置できるわけではない．利用者が端末として直接利用するようなノードでは，ユーザインタフェース処理などの非クラウド処理にもリソースを割く必要がある．そこで，我々が開発しているサーバレスクラウド基盤ソフトウェアでは，非クラウド処理に用いるリソースを予約し，クラウド処理に使えるリソース情報だけをクラスタコントローラに送信するようなリソース管理機構をノードコントローラに導入する．以下では，便宜上，クラウド処理のみを実行するコンピュータをサーバ，非クラウド処理を実行するコンピュータをクライアントと呼ぶ．

3.1 必要な機能

本リソース管理機構では以下のような機能が必要となる．

端末の種類の識別
: ノードコントローラとして動作しているサーバは，ほぼすべてのリソースをクラウド処理に利用可能だが，クライアントは一部のリソースを非クラウド処理用に割り当てる必要がある．そこでサーバとクライアントを識別する情報を取得する．

リソース情報の取得
: クライアントに使用される端末の性能はそれぞれが異なると考えられる．クライアントの性能によって非クラウド処理を行うために必要なリソース量は変化する．そこで，リソースの詳細な情報を取得する．

非クラウド処理用リソースの予約
: クラウド処理を実行させた時に，非クラウド処理の妨げにならない程度に非クラウド処理用リソースを予約する．また，非クラウド処理は常に実行中であるとは限らない．非クラウド処理が実行されていない，あるいは負荷が軽い場合，クラウド処理にも十分なリソースを割り当てる必要がある．そこで，非クラウド処理の状態によって非クラウド処理用に予約するリソース量を変化させる．

3.2 従来のノードコントローラの動作

ノードコントローラは，起動時にノードコントローラが動作しているコンピュータのリソース情報を取得し，クラスタコントローラへ送信する．これを受け，クラスタコントローラは各ノードで仮想マシンを作成可能かどうかを判断し，クラウドコントローラからの要求に従い仮想マシンの動作要求をノードコントローラに出す．クラスタコントローラから仮想マシンの作成要求を受けたノードコントローラは，ストレージコントローラから受け取った仮想マシンイメージをキャッシュして，ノード上の仮想化ハイパーバイザを利用して，仮想マシンを作成する．なお，作成した仮想マシンはクラスタコントローラからの要求により，再起動やスリープ，シャットダウンの動作を行うことができる．ノードコントローラは仮想マシンの動作状態や動作中の仮想マシンのリソース情報を定期的に取得し，クラスタコントローラへ送信する．

3.3 サーバレスクラウド基盤ソフトウェアにおけるノードコントローラ

ノードコントローラが起動する段階で，自分が起動しているのがサーバなのかクライアントなのかを判別する情報を作成する．また，クライアントの場合は端末の種類を判別した情報を取得する．次に，定期的にコンピュータの性能を比較できる情報を取得する．そして，クライアントと判別された場合は，端末の種類に合わせて非クラウド処理用のリソースを予約し，その他をクラウド処理に使用する．クライアントで利用者が非クラウド処理を行う場合は，非クラウド処理の負荷状況によってリソースの予約量を変化させる．これにより，非クラウド処理への影響を抑えつつ，非クラウド処理とクラウド処理を並行して実行できる．

4 システムの実装

3.3 節のノードコントローラの実装について述べる．なお，実装のベースとして Eucalyptus-3.2.0 を利用した．

4.1 端末の種類の識別

クライアントでは非クラウド処理用のリソースを予約する必要があるので，設定ファイルにサーバとクライアントを決める記述を追加することでサーバとクライアントを判別する処理を行う．また，クライアントとして使用される端末はそれぞれ性能が異なる．性能の違いをラップトップとデスクトップに大別し設定ファイルにラップトップとデスクトップを決める記述を追加する．ユーザは使用前にノードコントローラの動作設定としてサーバかクライアントかを記述し，クライアントの場合はラップトップかデスクトップかを記述する．

4.2 リソース情報の取得

既存のノードコントローラでは，仮想マシン作成時に使用するコンピュータの CPU コア数，総メモリ量が取得され利用されている．本システムでは，既存の情報に加

え，非クラウド処理の CPU 使用率を追加で取得する．非クラウド処理の CPU 使用率は，全体の CPU 使用率とクラウド処理の CPU 使用率の差を求めることで算出する．クラウド処理の CPU 使用率は仮想マシンが使用している CPU 使用率とした．我々は実際に仮想マシンを作成して CPU に負荷をかけた時に，ハイパーバイザである KVM プロセスが使用する CPU 使用率と仮想マシンが使用する CPU 使用率が同じであることを確認した．そのため，クラウド処理の CPU 使用率には，KVM プロセスの CPU 使用率を用いた．

4.3 非クラウド処理用リソースの予約

非クラウド処理用にリソースを予約する処理を行う．既存のノードコントローラは起動時にコンピュータの総メモリ量と CPU コア数を取得し，クラスタコントローラへ送ることで作成可能な仮想マシンの数を決める．そのため，リソースの予約には仮想マシンが利用可能なリソースに制限を設ければよい．今回は，比較的調整が容易であるクラスタコントローラに送るメモリ量を調整することでリソースの予約を行う．

クラスタコントローラへ送信するリソース量をあらかじめ減らす動作を行うことで作成できる仮想マシン数が制限でき，非クラウド処理用のリソース予約が可能になる．クライアントで行う非クラウド処理はユーザインタフェースを利用したもので GUI で動作するオペレーティングシステムを使用する．我々はオペレーティングシステムのバックグラウンドプロセスが常に使用するリソース量を調査し，結果から常に必要であるメモリ量 512MB の予約を行うことにした．

4.4 リソースの動的予約

クライアントは常に非クラウド処理を行っているわけではない．クライアントにおいては非クラウド処理をクラウド処理より優先する必要があるので，非クラウド処理の負荷が上がった時に非クラウド処理に割り当てるリソースを増やす．

我々は，非クラウド処理に掛かる負荷とメモリ使用量の関係を調べるため，1 アプリケーションが使用するメモリ使用量を測定した．図 3 と図 4 はラップトップとデスクトップ上で一定時間ごとに 1 つずつ様々なアプリケーションを実行し，結果をグラフにしたものである．横軸は経過時間，縦軸はそれぞれ CPU 使用率とメモリ使用量である．図 3,4 から，1 アプリケーションあたりの最大メモリ使用量は 1GB から 1.5GB 程度であることが分かる．

以上の結果から，プロセスの多重度が上がって CPU 使用率が 25% および 50% に達した時の必要メモリ量を予測し，非クラウド処理に割り当てるメモリ量を総メモリ量を元に表 1 の様に決定した．デスクトップとラップトップで搭載メモリ量が異なるのでそれぞれで異なる基準を適用した．

バックグラウンドプロセスに割り当てるメモリ量に合わせ，非クラウド処理に使用するメモリ量の動的予約処理を加えたものが図 5 である．バックグラウンドプロセスに割り当てたメモリ量は不変であるが，非クラウド処理に使用するメモリ量は非クラウド処理の負荷に合わせて変化する．

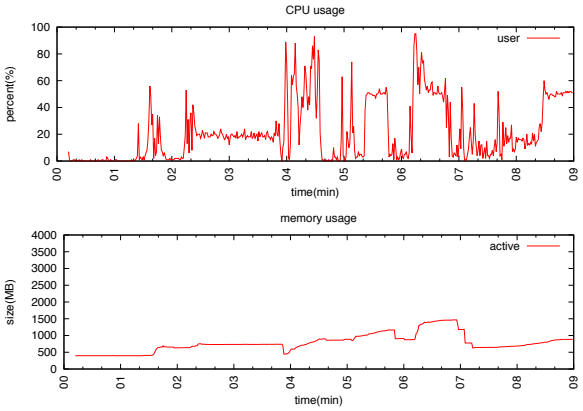

図 3 ラップトップ使用時の各リソースの使用率

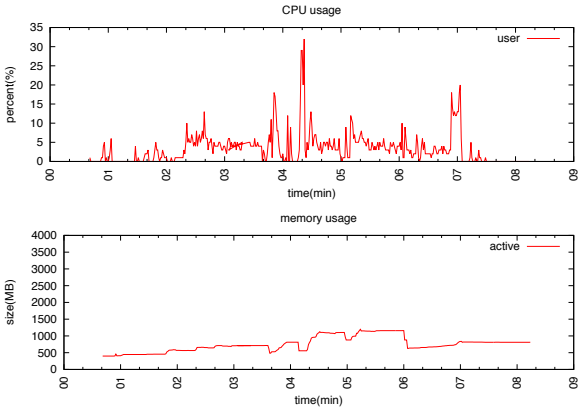

図 4 デスクトップ使用時の各リソースの使用率

非クラウド処理の CPU 使用率	予約するメモリ量 (ラップトップ)	予約するメモリ量 (デスクトップ)
0%〜25%	総メモリ量の 1/8	総メモリ量の 1/12
25%〜50%	総メモリ量の 1/4	総メモリ量の 1/6
50%〜	総メモリ量の 1/2	総メモリ量の 1/3

表 1 変化させるリソース量

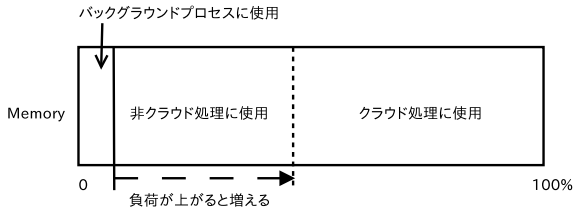

図 5 リソース予約機構

5 実験

クライアントで非クラウド処理を実行中にクラウド処理を行った場合に非クラウド処理が安定して実行できるか確認する実験を行った．

5.1 実験環境

本実験で使用したコンピュータの仕様を表 2 に示す．

	NC(デスクトップ)
CPU	Intel®Core™i7-2600
コア数	8
クロック周波数	3.40Ghz
メモリ	8GB
HDD	500GB
OS	ubuntu 12.04 Desktop

表 2 コンピュータの仕様

5.2 実験方法

既存のノードコントローラと本リソース管理機構を実装したノードコントローラをそれぞれ導入したデスクトップコンピュータを用いて非クラウド処理の処理速度の比較を行った．非クラウド処理を実行中にクラウド処理を行った場合を想定して，今回はあらかじめ非クラウド処理にかかる負荷が 50% 以上の状態を繰り返し計算を行うプログラムで作成しておき，クラウド処理を開始する．この時仮想マシンを最大まで作成し，仮想マシンでは処理の負荷が最高に掛かっている状態を作成するために大量の配列を利用するプログラムでメモリを十分に使用する．その後非クラウド処理として 1 時間の音声ファイルのエンコードを行い開始から終了までの時間を計測する．計測はそれぞれのノードコントローラで 5 回行い結果を比較した．

5.3 結果

表 3 は既存のノードコントローラと本リソース管理機構を実装したノードコントローラでの実験結果である．

	既存 (仮想マシン数:3 台)	本システム (仮想マシン数:2 台)
1 回目	94.352924	88.811337
2 回目	93.003637	89.96591
3 回目	93.349833	89.83812
4 回目	92.275797	90.642548
5 回目	91.683260	85.169757
平均	92.93309	88.88553

表 3 処理時間 (秒)

結果から，本リソース管理機構を実装したノードコントローラの方が処理速度が速いことが分かる．非クラウド処理に使用できるメモリ量が増えたので，既存のノードコントローラと比較すると非クラウド処理の処理速度が速くなったと考えられる．一方，非クラウド処理とクラウド処理を同時に実行した場合に，本システムではクラウド処理に使用できるメモリ量が減るので，作成できる仮想マシン数は 3 台から 2 台へ低下している．

今回はエンコードを行うことで比較を行ったが，更に大きな処理や多くのメモリを使用する処理を行った場合に結果が大きく異なってくると考えられる．処理速度が変わることでソフトウェアの立ち上げなどの体感速度も変わり，非クラウド処理中にクラウド処理を行っても非クラウド処理の妨げになるのを防ぐことができると言える．

6 関連研究

既存のオープンソースクラウド基盤ソフトウェアには，Eucalyptus[3]，Open Stack[4]，Cloud Stack[5] などがあるが，いずれもクライアントコンピュータをノードとして使用することを想定していない．

7 まとめと今後の課題

我々は，サーバレスクラウド基盤ソフトウェアのノードコントローラにおけるリソース管理機構の開発を行った．非クラウド処理用にリソース予約を行うことにより，クラウド処理が非クラウド処理の妨げになることを防ぐことができクライアントをクラウドのノードとして利用できるようになる．

本システムの有効性を調べるために，本リソース管理機構の有無による非クラウド処理の処理速度の比較を行った．その結果，非クラウド処理中に同時にクラウド処理を実行した場合に本リソース機構を用いることでクラウド処理が非クラウド処理の妨げになるのを防ぐことができることを確認した．

今後の課題として，起動中の仮想マシンに割り当てたリソース量をノードコントローラが自由に変更できるように改良する必要がある．

参考文献

[1] 加藤英雄, “クラウドコンピューティング-サーバは雲のかなた,” 2011.

[2] 青木勇貴，長谷川浩之, “プライベートクラウド環境における分散ファイルシステムの試作,” 2012 年度南山大学情報理工学部ソフトウェア工学科卒業論文 2013.

[3] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The eucalyptus open-source cloud-computing system,” *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'09)*, 2009.

[4] Apache License 2.0, “Open source software for building private and public clouds..” http://www.openstack.org/.

[5] The Apache Software Foundation, “Apache CloudStack.” http://cloudstack.apache.org/.