Vol.2011-HPC-130 No.71
2011/7/29

ローカルクラスタ環境のクラウドへの移行機構

孫コウ[†]　　合田憲人[††]

高性能計算分野においてIaaSクラウドの利用に関して研究が多くなされている．従来研究においてIaaSクラウドの性能評価がなされたが，本論文ではクラウドを用いたローカルクラスタ環境をクラウドへ移行するためのシステムを検討する．本論文では移行機構の要求要件をまとめて，プロトタイプシステム InterS を実装し環境移行の実現可能性を示す．NPB を用いて環境移行の性能を評価し，予備実験の結果から InterS システムのオーバヘッドは約 13 分であることが分かり，ローカルクラスタの環境移行の可能性を示した．

Migration Mechanism Between Local Cluster and Public Cloud Environment

Hao SUN[†]　and　Kento AIDA[†,††]

In High Performance Computing many researches have be done on IaaS Cloud utilization. Some of them evaluated IaaS Cloud performance for scientific applications, some of them extended local cluster with Cloud resources to guarantee QoS of the entire system. In this paper, we propose a mechanism to automatically migrate local cluster environment to IaaS Clouds and the requirements to build such a system is shown and a prototype system called InterS is built. To evaluate performance, NAS paralell benchmark EP program is used. From the preliminary experiment results, overhead of migrate entire cluster enviroment is about 13 minutes and finally all tasks finished successfully within the new IaaS Cloud environment.

1. はじめに

近年科学技術計算分野においてIaaSクラウドの活用が注目されるようになった．ローカルクラスタと IaaS クラウドをハイブリッドして利用することで計算資源の不足を補ってQoS（Quality of Service）を保証できるほか，仮想マシンの集約による省電力計算においても注目されている．さらに，IaaSクラウドの計算資源の分散配置を利用し，ハザードディザスターにも利用可能である．例えば，2011年日本大震災の影響で計画停電が実施され，ローカルクラスタ環境をIaaSクラウドへシームレスに移行できれば，タスクが継続実行されシステムのQoSを保証することができる．

これまで高性能計算分野においてIaaSクラウドの利用，あるいはローカルクラスタとハイブリッドして利用する提案が多くなされているが，ローカルクラスタ環境をそのままクラウドへ移行する方法論はまだ確立されていない．関連研究としてローカルクラスタ環境構築ツールとして有名な Lucie[1]，プライベートクラウドにおいて仮想マシンのマイグレーション機構などが挙げられるが，これらの研究を利用してもローカルクラスタ環境をシームレスに IaaS クラウドへ移行することはできない．例えば，クラスタ環境構築ツールの多くはローカルネットワークを想定し作られているためIaaSクラウドへの応用が難しい．仮想マシンのマイグレーション機構は仮想マシンの使用を前提にしているため，仮想化されていないローカルクラスタに対応できない．さらに，仮想マシンのマイグレーション機構は主に移行時の性能に研究がなさてれいるため，異種クラウド間のネットワーク構造差異の吸収や移行後システムの全体性能の保証などを問題として挙げられていない．

ローカルクラスタ環境を必要に応じてIaaSクラウドへ移行するツールInterSを作成し，その利用によりユーザは以下のメリットを得ることができる．

(1) 必要に応じて IaaS クラウドの資源を利用できる．例えば，デッドラインに間に合わせるためにIaaSクラウドの資源を一時的に借りることができる．
(2) 電力利用効率の向上が見込める．データセンタにある計算資源の電力利用率がローカルクラスタよりよい傾向が見られる．例えば，Facebook のデータセンタがOpenCompute[2]コンセプト適応し電力消耗が38%削減できている．仮想マシンの集約による電力削減の研究が注目さてれいるなか，将来，IaaSクラウドのデータセンタにおいて応用される見込みもあるため，積極的にクラウドの利用によりグリーンコンピューティングの実現に貢献できる．
(3) 災害の影響を回避する手段として利用できる．例えば，2011 年 3 月の日本大震

† 国立情報学研究所
National Institute of Informatics
†† 東京工業大学
Tokyo Institute of Technology

災により計画停電の実施が余儀なくとされている現在，ローカルクラスタ環境を停電前にシームレスに IaaS クラウドへと移行できるとタスクの実行が継続され，システムの QoS を保証することができる．

本稿ではローカルクラスタ環境を移行するために，ローカルクラスタの設定を IaaS クラウドにおいてを再現する方法を検討し，プロトタイプのシステム InterS を実装し性能測定を行う．クラスタの設定を抽出管理するためのツールとして blueprint[3]，cft（Configure File Tracking）[4]などが有名であり，本研究ではより広範囲に適用できるツールとして blueprint を利用する．ローカルクラスタの設定を自動抽出においてはまだ問題点多く残されているが，本論文ではローカル・バッチスケジューラ環境の再現とタスクの継続実行を問題として定め，その方法論について議論する．

以下，第 2 節で関連研究について紹介したあと，第 3 節でシステムを実現するための要求要件をまとめる．第 4 節では InterS システムの実装について述べ，第 5 節で予備実験を通してシステムの性能を示す．最後，第 6 節では本論文をまとめて将来の課題について述べる．

2. 関連研究

関連研究として(1)ローカルクラスタ環境構築ツール，(2)プライベートクラウドにおいて仮想マシンのマイグレーション機構，(3)クラスタ環境設定の抽出ツールとして挙げられる．それぞれについては以下で詳述する．

(1) ローカルクラスタ環境構築ツール

大規模クラスタ構築ツールとして OSCAR[5]（Open Source Cluster Application Resources），Rocks[6]，Lucie などが挙げられる．これらのツールはローカル環境においてクラスタを構築する際大量の計算ノードへのインストールタスクを簡単化し，自動管理してくれる．

(2) プライベートクラウドにおいて仮想マシンのマイグレーション機構

仮想マシンのライブマイグレーション機構について様々な研究がなされている．LAN において仮想マシンを移動するツールとして産総研の瞬間実行ホスト切り替え高速ライブマイグレーション機構[10]，SnowFlock[7]などが挙げられる．両方とも Xen の仮想マシンモニタ(VMM)を改造して仮想マシンの移動効率を工夫し，1 秒以下までオーバーヘッドを削減できている．そのほか，WAN/MAN においては仮想マシンを移動する研究もある[8]．

(3) クラスタ環境設定の抽出ツール

一台の計算機の設定を抽出するツールとして blueprint，cft(Configure File Tracking)などがある．blueprint はシステムの様々なパッケージ管理システム（apt-get，yum，pear など）からパッケージのリストとそのバーションを抽出し再インストールするためのスクリプトを生成できる．cft は RedHat 系列のシステムの変更を監視しそれらの変更を再現できる．関連研究として Liang ら[9]により Linux，Unix-like なシステムの設定ファイルを管理するために Service Hosting Environment モデルが提唱され，IBM の製品により実用化されている．

しかし，これらの関連研究はローカルクラスタ環境をシームレスに IaaS クラウドへ移行することはできない．クラスタ環境構築ツールの多くはローカルネットワークを想定し作られているため，広域ネットワークにある IaaS クラウドへの応用が難しい．仮想マシンのマイグレーション機構は仮想マシンの使用を前提にしているため，仮想化されていないローカルクラスタと Amazon EC2 などのパブリッククラウドに対応できていない．さらに，仮想マシンのマイグレーション機構は主に移行時の性能に研究がなさてれいるため，異種クラウド間のネットワーク構造差異の吸収や移行後システムの全体性能の保証などを問題として挙げられていない．なお，これらの研究は大きいサイズのファイルの移動に関して多くのテクニックが使用されているため，それらは本研究においても適用できる．計算機設定の抽出に関しては既存ツールと研究の成果を取り入れている，現在の実装では設定を抽出するには blueprint ツールを利用している．

3. 検討システムへの要求要件

本研究ではローカルクラスタ環境の設定をクラウドへ適用し環境の再現をするアプローチを取っている．ローカルクラスタ環境のタスクがクラウド環境で正常実行する為に，以下の要求要件を満たす必要があると考えられる．なお，クラウドにおいてシステム全体の QoS を管理する機能が別当必要とし今後の課題として挙げられる．IaaS クラウドとして Amazon EC2 を利用する．

(1) クラスタの構成の再現

（ア）ネットワークのレイアウトの再現
IP アドレスが変更されてもタスクの正常実行に影響を与えないと仮定する．タスクに IP アドレスがハードコーディングされ，変更に対応できない場合は考慮しない．

（イ）ネットワークの通信とその性能の再現

① Amazon EC2 は multicast と broadcast がサポートしていないので再現しがたい．

② MPI タスクは Amazon EC2 のクラスタインスタンスを利用する．

③ ローカル環境は Infiniband が利用しているクラスタは Amazon EC2 の

クラスタインスタンスを利用してより近い性能を得る．

（ウ）ハードウェアの再現

① ハードディスク容量を再現する．

② ハードディスクレイアウトを再現する．

③ NFS，iSCSI などの共有ストレージを再現する．

④ GPU クラスタは Amazon EC2 の GPU インスタンスを利用する．

（エ）計算ノードの再現

システム全体の計算性能と同性能になるようにクラウドの計算資源を選択する．

(2) バッチスケジューラ設定の再現

（ア）すべてのタスク正しく実行されること

① ローカルクラスタ環境で実行中のタスクはクラウド環境で再実行される．

② ローカルクラスタ環境で実行まちのタスクはクラウド環境で実行される．

（イ）キュー設定の再現

キューの各種設定を再現する必要がある．例えば，キューに含まれるノード，アクセス制御，実行締切りの設定などが挙げられる．

（ウ）計算ノードの再現

バッチスケジューラに設定されたプロパティを再現する必要がある．例として Torque では server_priv/nodes に書かれているものがある．

(3) ユーザの情報の再現

ユーザの情報は何らかの方法により再現できると仮定する．例えば，現在の実装では/etc/passwd から情報を抽出している．NIS，LDAP などにより管理されている場合は別当対応する必要がある．

(4) タスク実行に関わる環境の再現

以上の要件以外は下記のことも満たす必要がある．

（ア）タスク実行に必要なパッケージの再現

① OS レベルのパッケージ： apt-get，yum あど

② 言語システムのパッケージ： gem，pip，pear，cpan など

（イ）独自ライブラリの再現

① システム全体に反映したもの：/etc/ld.so.conf.d/にあるもの

② ユーザ別に設定したもの：LD_LIBRARY_PATH に設定されるもの

（ウ）タスクのインプット・アウトプットデータ

① qsub で指定されるもの

② タスクファイルの書かれたもの

4. システムの設計と実装

本稿で述べている InterS の実装はクラウドにおいてクラスタ環境の再構成とタスク移行の方法論に中心としているため，第 3 節で挙げられた要件の内最低限必要となる一部を実装した．具体的に（1）のア，（2）のア，イ，ウ，（3）と（4）のウが実装されている．将来，その他の要件についても実装する予定である．

4.1 システムの全体構成

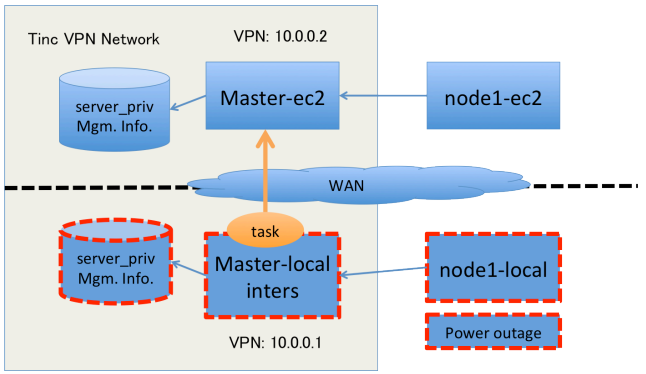

図 1 ローカルクラスタ環境のマイグレーション

ローカルクラスタ環境を IaaS クラウドへマイグレーションする全体図として図 1 を用いて説明する．図 1 はローカルクラスタ環境（点線で示した下半分），Amazon EC2 クラウド環境（上半分）と WAN により構成されている．提案システム InterS がローカルクラスタの Master-local に動いている．ユーザは表 1 に示さるスクリプトを用いてローカルクラスタ環境をクラウドへ移動できる．将来，一度 IaaS クラウドへ移動したクラスタ環境をローカルクラスタへ戻すためのスクリプトはまだ実装されていないが作成する予定がある．

表 1 InterS を利用するためのスクリプト

スクリプト名	役割
create-my-cloud	ローカルクラスタ環境の情報を収集し，同じ構成のクラスタ環境を IaaS クラウド上で構成する．ローカルで実行されているタスクをクラウドへと移動され，実行される．
stop-my-cloud	タスクの実行結果を収集して IaaS クラウドにあるクラスタを消す．

Master-local はローカルクラスタ環境でのバッチスケジューラのマスターノードであり，node1-local は計算ノードの一例を示している．本論文ではバッチスケジューラとして広く利用されている Torque をベースにシステムを設計しているため，Master-local には pbs_server, pbs_sched プロセス，node1-local には pbs_mom プロセスが動作している．Master-local が参照している server_priv にはバッチスケジューラ全体の管理情報，例えば，キューの種類，ジョブの状態，計算ノードの情報などが格納されている．Amazon EC2 クラウドにある Master-ec2, node1-ec2 と server_priv がローカルクラスタ環境と同じ機能を果たし，ローカルクラスタは計画停電などの原因によりタスクを Amazon EC2 クラウドへ移動する需要に応じて，InterS により動的に作成される．

InterS はタスクの移動の際，データ内容を VPN を用いて保護する．具体的に Tinc というオープンソースソフトウェアを利用して Master-local と Master-ec2 の間に 10.0.0.x のネットワークを動的に形成後 server_priv のデータを Master-local から Master-ec2 へ複製する．TincVPN ネットワークの形成，Master-local から Master-ec2 への複製は互いに独立されているタスクであるため，cron を用いて実装している（4.20 参照）．server_priv のデータがアップロードされてから，Master-ec2 にある pbs_server が cron ジョブにより再起動され，管理情報の更新が反映されてタスクがクラウド環境で実行される．現時点では，InterS がジョブのチェックポイントをせずにタスクを移動するため，ローカルクラスタで実行中のタスクはクラウド環境において再実行される．ローカルクラスタで実行待ちタスクがクラウド環境の Master-ec2 により資源配分され実行される．

4.2 クラスタ環境の移行手順

InterS が動的にクラスタ環境を作成する手順を以下で詳述する．

(1) ローカルクラスタ環境設定の収集

本研究では IaaS クラウド上でクラスタの環境を構築する時ローカルクラスタ環境の設定を用いるため，ローカルクラスタの環境設定を収集する必要がある．設計上 InterS はすべての情報の自動収集を目指しているが，本論文では以下の設定をそれぞれ自動収集と事前設定を用いることとする．なお，現実装では一部対応していない設定もある．

- 自動収集する情報：Torque の管理情報（キュー，ユーザ，計算ノードに関する設定），クラスタのユーザ情報（ユーザ名，所属するグループ），インストールされたパッケージ情報（yum，apt-get，rubygems，pip，pear によりインストールされたパッケージ名とそのバージョン），動的ロードされるリンク情報（/etc/ld.so.conf.d/）
- ユーザが事前設定する情報：ユーザ独自コンパイルしたバイナリの場所，その依存ライブラリ（/etc/ld.so.conf.d/に設定がない場合）の情報，パッケージ管理システム以外のパッケージがインストールされる場所（デフォルトでは/usr/include, /opt として設定される）．将来，ユーザ独自コンパイルしたバイナリの依存関係も ldd などのコマンドにより自動収集する予定がある．
- 未対応の設定：ハードディスクのレイアウトと同容量の確保，GPU の有無（Amazon EC2 の GPU インスタンスの利用），MPI タスクの判定（Amazon EC2 のクラスタインスタンスの利用），NFS の設定，autofs の設定，ユーザアカウント管理システムへの対応（例えば NIS や LDAP の設定）

(2) IaaS クラウドにおいてクラスタの作成

図 2 を用いて Amazon EC2 クラウド上でクラスタを作成する時の概要を示す．InterS は表 1 で示した install-ec2.local-task と install-ec2.ec2-task 二種類のタスクを用いて IaaS クラウドクラスタを作成する．具体的に install-ec2.local-task は Amazon EC2 のインスタンスを生成し，ローカルクラスタ環境の設定ファイルをアップロードして install-ec2.ec2-task を呼び出す．install-ec2.ec2-task は実行環境設定管理システム Puppet をインストールしたあとアップロードされた設定適用し，Torque システムの正常動作を確認する．install-ec2.local-task は InterS がローカルマシンつまり Master-local で行うタスクであり，install-ec2.ec2-task は各クラウド上の仮想マシンで行うタスクである．

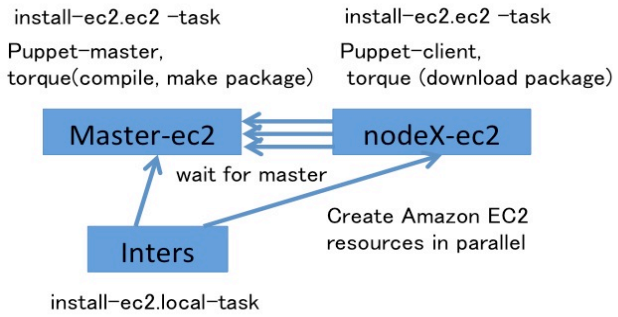

図 2　Amazon EC2 でのクラスタ作成

InterS はクラウドの計算資源を確保する時 install-ec2.local-task は計算ノードの数に比例して重くなり Master-local の CPU 資源を多く消耗する可能性がある．そのため，Master-local マシンへの負荷を軽減するために現実装ではできるたけ多くのタスクを install-ec2.ec2-task へ割り当てるように設計されている．install-ec2.ec2-task は各クラウドの仮想マシンにおいて実行されるためほぼ一定の時間で実行終了するが，Amazon EC2 の API 呼び出しやパッケージのダウンロード，インストールはネットワークと計

算機の性能により変動する．なお，install-ec2.local-task の処理は Master-ec2 と nodeX-ec2 を作成時実行される回数が異なるため，実行時間も違ってくる．一部のタスクは Master-ec2 の作成時だけ実行される（表 2 実行回数参照）. install-ec2.local-task と install-ec2.ec2-task の処理性能関しては第 5 節で示す.

表 2 IaaS クラウドクラスタ作成するためのタスク一覧

タスク名	分類	役割	実行回数
1.add_ec2_key	install-ec2.local-task	Amazon EC2 への SSH キーの生成	1 回(マスターだけ)
2.create_ec2	install-ec2.local-task	Amazon EC2 インスタンスの作成	n 回(クラスタノード数)
3.allocate_eip	install-ec2.local-task	Amazon EC2 の ElasticIP 作成と Master-ec2 インスタンスへの割り当て	1 回(マスターだけ)
4.open_ports	install-ec2.local-task	Amazon EC2 の firewall に SSH, TincVPN のポートを開ける	1 回(マスターだけ)
5.ssh_config	install-ec2.local-task	SSH アクセスするための設定ファイル.ssh/config を作成する	n 回(クラスタノード数)
6.add_tinc_cron	install-ec2.local-task	TincVPN への接続をテストするための cron ジョブを作成，登録（0 へ参照）	1 回(マスターだけ)
7.add_ec2_hosts	install-ec2.local-task	Amazon EC2 の IP と hostname を/etc/hosts へ登録	n 回(クラスタノード数)
8.add_torque_cron	install-ec2.local-task	Torque タスクを移行するための cron ジョブを作成，登録（0 へ参照）	1 回(マスターだけ)
9.install	install-ec2.local-task	クラスタ設定ファイルのアップロードと install-ec2.ec2-task の呼び出し	n 回(クラスタノード数)
1.prepare	install-ec2.ec2-task	ruby, git など InterS 依存するパッケージをインストール	各ノードに 1 回
2.add_ec2_tags	install-ec2.ec2-task	stop-my-cloud の実行時，計算ノードの情報を獲得するためのタグが Amazon EC2 API を利用して付けられる.	各ノードに 1 回
3.set_puppetenv	install-ec2.ec2-task	実行環境管理システム Puppet をインストールするための事前設定	各ノードに 1 回
4.add_cronjobs	install-ec2.ec2-task	Puppet クライアントがマスターへ接続するための cron ジョブの作成と登録	各ノードに 1 回
5.install_puppet	install-ec2.ec2-task	Puppet システムのインストール	各ノードに 1 回
6.apply_settings	install-ec2.ec2-task	Puppet によりアップロードされたクラスタの設定を適用する	各ノードに 1 回

(3) ローカルクラスタにあるタスクの移動

InterS は Master-ec2 の構築完了後ローカルクラスタにあるタスクをクラウドへと移動する．Torque システムでは Master-local において server_priv ディレクトリにすべて管理情報が置かれているため，このディレクトリのデータを複製すればタスクの実行に関わる情報を揃えることができる．ただし，すでに実行中のタスクに関しては実行 server_priv ディレクトリには実行ノードの情報も含めているため，複製するだけでは継続実行はできない．問題を回避するために，現実装では，まず pbsnodes –o コマンドですべての計算ノードを offline 状態へと切り替え，その後 qrerun コマンドを利用して実行中のタスクを再び実行待ち状態へ戻している．qrerun を利用すれば Torque により当てられた JOBID が変更されずにタスクがキューイング状態へ戻すことができる．

なお，タスク移動は Master-local において，かつ Master-ec2 の構築完了後で実行される必要があるため，InterS は cron ジョブを利用して実装を行った．add_tinc_cron と add_torque_cron タスクを用いて Master-ec2 の構築が完了したかを 1 分置きにチェックする．Master-ec2 の構築が完了したら add_tinc_cron は Master-local と Master-ec2 の間の TincVPN の接続の疎通をテストし，その後 add_torque_cron は Master-local にあるタスク情報を Master-ec2 へとコピーし，install-ec2.ec2-task の add_cronjobs で登録された cron ジョブにより Amazon EC2 クラウドにある Torque (pbs_server)が再起動される．よって，アップロードされた server_priv ディレクトリの情報が反映され，キューにあるタスクがスケジュールされ，クラウドのクラスタにおいて実行される．なお，add_tinc_cron と add_torque_cron は成功したあと自動的に crontab から削除されるため，システムへの負荷はすくなくで済む.

5. 性能評価

InterS によるローカルクラスタ環境の移行を評価するために，(1) 移行にかかる時間と (2) 移行後タスクが正常に実行されたかを確認する予備実験を行った．

実験では Amazon EC2 の東京リージョンと国立情報学研究所にある Torque (Version 2.5.6) に管理された 5 台構成のローカルクラスタを使用した．ローカルクラスタでは各ノードが CPU(Dual-Core AMD Opteron(tm) Processor 2216, 2.4GHz)2 個，OS として Ubutnu Lucid 10.04 の 32bit がインストールされている．Amazon EC2 のインスタンスとしては同じ OS で 32bit の c1.medium インスタンス 10 個 (インスタンス毎に 2.5GHz の仮想コア 2 個) を利用した．プログラムとして NPB (NAS Parallel Benchmark 2.4-OPM-C) の EP プログラム Class C，100 タスクをローカルクラスタのキューに投入した．一個の EP タスクはローカルマシンと c1.medium インスタンスでの実行時間はそれぞれ約 927(s), 900(s)である．EP プログラムはローカル環境のライブラリに依存しないタスクであるため InterS によりクラウド環境にコピーされて実行できた．ライブラリの依存関係のあるタスクに関しては，ユーザにより依存関係を示す必要があるため，今後対応する必要がある．

実行のシナリオとしては，ローカルクラスタで EP プログラムを投入直後，Master-local において InterS の create-my-cloud スクリプトを実行し，Amazon EC2 においてクラスタを作成した．以下では環境作成にかかる時間とタスク移行の実行結果をそれぞれ詳述する．

5.1 IaaS クラウドにおいてクラスタ環境構成にかかった時間

クラウドにおいてクラスタ構築にかかる時間として install-ec2.local-task の処理はマスタと計算ノードの場合ではそれぞれ 91(s)と 42(s)かかり，install-ec2.ec2-task 処理は 720(s)と 177(s)かかった．よって，マスタの構築時間はおよそ 800(s)，計算ノードの構築時間はおよそ 200(s)かかることがわかる．各タスクにかかる時間の詳細は表 3 にまとめられている．なお，apply_settings はローカルクラスタ環境の適用にかかる時間であるため，実験環境により大幅変動すると予想できる．

クラウド上のマスタと計算ノードの構築に時間差大きくあった．原因として Torque と Puppet システムの構築にある．本研究ではローカル環境と同じ torque システムを構築するために OS に用意されたパッケージを利用せずに Amazon EC2 上のマスタノードにおいて新たにパッケージを作成し，Puppet を通して計算ノードへ配布・インストールしている．よって，Amazon EC2 の Torque マスタノードの構築時間は puppet のマスタノードの構築時間と torque をソースコードから構築する時間が含んでいて，計算ノードより時間がかかる．表 3 から分かるが install_puppet，apply_settings タスクはマスターにおいてそれぞれ 120(s), 343(s)長くなる．

InterS はマスタと計算ノードの構築を同時開始したため，IaaS クラウドでの環境構築の全体時間はマスタノードの構築時間と同じであり，この実験においてはおよそ 800(s)であることが分かる．

表 3　各タスクでクラスタ環境構成にかかった時間

タスク名	種類	実行時間 (s)
1.add_ec2_key	install-ec2.local-task	18
2.create_ec2	install-ec2.local-task	39
3.allocate_eip	install-ec2.local-task	15
4.open_ports	install-ec2.local-task	15
5.ssh_config	install-ec2.local-task	1
6.add_tinc_cron	install-ec2.local-task	1
7.add_ec2_hosts	install-ec2.local-task	1
8.add_torque_cron	install-ec2.local-task	1
9.install	install-ec2.local-task	1
1.prepare	install-ec2.ec2-task	17
2.add_ec2_tags	install-ec2.ec2-task	3
3.set_puppetenv	install-ec2.ec2-task	1
4.add_cronjobs	install-ec2.ec2-task	1
5.install_puppet	install-ec2.ec2-task	Master: 300, node: 80
6.apply_settings	install-ec2.ec2-task	Master: 418, node: 75

5.2 クラウド環境に移動されたタスクの実行結果

この実験ではタスクがクラウド環境へ移動される様子を示す．シナリオとしてはユーザが 100 個の EP プログラムをローカルクラスタのキューへ投入直後，InterS の create-my-cloud スクリプトを用いてクラウド環境を構築し (開始時間 09:34:01)，タスクの移行 (開始時間 09:47:02) を行った．

表 4 で示した結果により 100 個のタスクが正しく実行されたことが確認できる．一つのタスクかかる時間は約 905(s)であり，クラウド環境の構築時間とタスク移行のオーバーヘッドはそれぞれ 780(s)と 130(s)となっている．本実験では Amazon EC2 の環

境はローカルクラスタ環境とほぼ同じ性能が発揮したことが分かる．

表 4　クラウド環境構築とタスク移行の履歴表

タイムスダンプ	タスク名	時間の差分(s)[注]
09:34:01	クラウド環境構築開始． create-my-cloud が実行された時刻	---------
09:47:02	InterS による Torque のタスク移動	780 [マスターノードの構築時間]
09:49:12	クラウド環境において最初のタスクの実行開始時刻（JOBID: 8786）	130 [タスクマイグレーション時間]
11:04:38	クラウド環境において最後のタスクの実行終了時刻（JOBID: 8885）	4526 [100 個タスクが 20 並列での実行時間]

本実験は InterS のシステム固有のオーバーヘッドを示されているが，ローカルクラスタからデータの移動が含まれていないため，実運用環境に適用される場合はもっと時間がかかることを予想される．

6. まとめ

本論文ではローカルクラスタ環境をクラウドへ移行するためのシステムを検討し，システムの要求要件をまとめて，プロトタイプシステム InterS を実装した．実験結果から分かるがバッチスケジューラの環境は移行できることが分かり，そのオーバヘッドは約 13 分であるため，システムの実用可能性を示唆している．

関連研究としては(1)ローカルクラスタ環境構築ツール, (2)プライベートクラウドにおいて仮想マシンのマイグレーション機構, (3)クラスタ環境設定の抽出ツールとして挙げられるが，ローカルクラスタ環境全体を移行できるツールはまだ存在しない．

今後の予定としては InterS の実装を改良してまだ満たされていない要求要件を対応し，より複雑なローカルクラスタ環境への適用実験が必要となる．

参考文献

1) Takamiya, Y., Manabe, A., SHIRASUNA, S., & MATSUOKA, S. Lucie : A Fast Installation and Administration Tool for Large-scaled Clusters. IPSJ Journal, p79-88, 2003.08
2) Open Compute Project. http://opencompute.org/ (2011.06)
3) Blueprint. https://devstructure.com/docs/tutorial.html (2011.06)
4) Cft. http://cft.et.redhat.com/ (2011.06)
5) OSCAR: Open Source Cluster Application Resources. http://svn.oscar.openclustergroup.org/trac/oscar (2011.06)
6) Rocks Clusters. http://www.rocksclusters.org/wordpress/ (2011.06)
7) Lagar-cavilla, H. A., Whitney, J. A., Scannell, A., Patchin, P., Rumble, S. M., Lara, E. D., et al. SnowFlock : Rapid Virtual Machine Cloning for Cloud Computing, Proceedings of the 4th ACM European conference on Computer systems (EuroSys 09). p1-12, 2009
8) Travostino, F., Daspit, P., Gommans, L., Jog, C., Delaat, C., Mambretti, J., et al. Seamless live migration of virtual machines over the MAN/WAN. Future Generation Computer Systems, 22(8), p901-907. 2006.03
9) Liu, L., Li, Y., Ma, Q., Sun, K. W., Chen, Y., & Wang, H. Automatic model-based service hosting environment migration. 2008 IEEE Network Operations and Management Symposium. p682-685. 2008.08
10) 広渕崇宏，中田秀基，伊藤智，関口智嗣,”瞬間的な実行ホスト切り替えを可能とする仮想マシンの高速ライブマイグレーション機構”，日本ソフトウェア科学会研究会資料シリーズ(62)，p57-66，2009.10